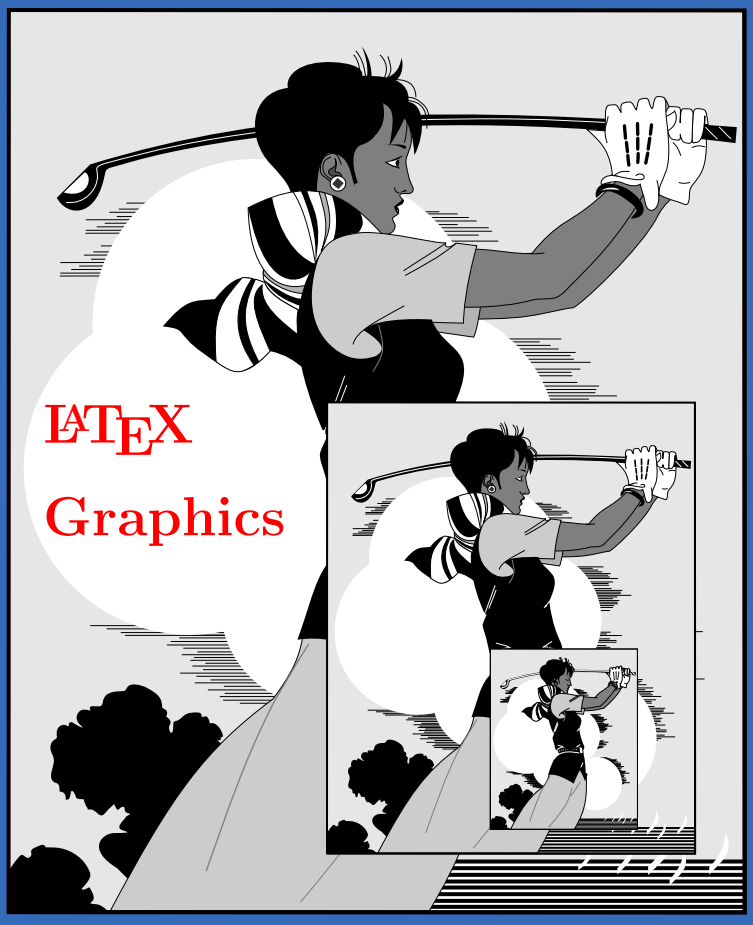
LaTeX 2ε

插图指南



Using Import graphics in LaTeX 2ε

Keith Reckdahl
reckdahl@am-sun2.stanford.edu

Version 2.0
December 15, 1997

前　言

本书是 Keith Reckdahl 所著的 “Using Imported Graphics In LaTeX 2$_\varepsilon$” 一文的中文译本，并加入少许译者自己的使用心得所成。在用 LaTeX 2$_\varepsilon$ 编写论文、书稿时，经常要遇到使用图形的情况，本书将讲述如何在 LaTeX 2$_\varepsilon$ 文件中插入图片以及与此有关的一些其它问题。由于完全阅读本书需要较多的时间，许多情况下您可以先浏览目录或索引，直接阅读您所需的相关内容。

尽管在 LaTeX 中能够试用几乎所有的图像格式，但 Encapsulated PostScript (EPS) 是最早被引入 LaTeX 中的图像格式，因此对它的支持也是最好的。例如，要在 LaTeX 文件中加入一幅 EPS 图像，可在全文设定区 (preamble) 中加入：

```
\usepackage{graphicx}
```

然后，在文件中用下面的命令来加入图形—`file.eps`

```
\includegraphics{file.eps}
```

还可加入 `height` 或 `width` 选项来使得所插入的图形缩放为指定的高度或宽度：

```
\includegraphics[height=4cm]{file.eps}
\includegraphics[width=3in]{file.eps}
```

另外，使用 `angle` 选项来旋转所插入的图形。

```
\includegraphics[angle=45]{file.eps}
```

有关 `\includegraphics` 的详细讨论以及 LaTeX 2$_\varepsilon$ 图形宏包的其它命令在本书的第二部分。

本书分为如下几个部分：

第一部分： 背景介绍

在这一部分中，介绍了一些有关的历史资料和基本的 LaTeX 术语。同时，也介绍了 Encapsulated PostScript (EPS) 图形格式，EPS 和 PS 的不同之处，以及将其它图形格式转为 EPS 格式的方法。

第二部分： LaTeX 2_ε 图形宏包套件

在这一部分中，详细介绍了 LaTeX 2_ε 图形宏包套件中用于引入、缩放和旋转图形的命令。这部分涵盖了 LaTeX 2_ε 图形宏包的文档的大部分内容（参见 [5] ）。

第三部分： LaTeX 2_ε 图形命令的使用

这一部分介绍了如何使用 LaTeX 2_ε 图形宏包套件中的命令来引入、缩放和旋转图形。此外还讨论了以下三种情况：

- 在支持管道（像 UNIX ）的系统中，使用 `dvips` 可以插入压缩的 EPS 图形或其它格式的图形（ TIFF, GIF, JPEG, PICT,etc. ）。在其它的系统中，非 EPS 格式的图形必须先转换为 EPS 才行。因为无论是 LaTeX 还是 `dvips` 都没有解压缩和转换图像格式的能力，所以使用者需要提供所需的软件。
- 由于许多应用程序支持 ASCII 文本，PSfrag 这一宏包可以将 EPS 图形中的文字替换为 LaTeX 符号或数学表达式。
- 当一个 EPS 图形被多次使用时（比如文字后面或页眉上的标记），最后生成的PostScript 文件会将此 EPS 图形包含多次，当所使用的图形不是位图格式时，可以通过定义一 PostScript 命令来避免此图形被重复插入，从而使得到的 PostScript 文件较小。

第四部分： LaTeX 2_ε 图形环境

将所要插入的图形放置于 LaTeX 2_ε 的图形环境（ figure ）中有很多好处，在图形环境中的图形会被自动编号，从而可被引用或加到目录中。因为置于图形环境中

的图形可以通过浮动来很好分页，所以可以很容易的制作出具有专业水准的文稿。

除了 LaTeX 2ε 图形环境的内容外，这一部分还讲述了如下和图形有关的一些内容。

- 怎样自定义图形环境，例如怎样调整图形的放置位置，调整图形周围的距离，标题的距离和在图形与文本之间加入横线等等。还可以自定义标题的格式，自由地改变标题的式样、宽度和字体。
- 怎样在竖排页面版式的文档中加入横排的图形？
- 怎样将标题放置于图形的两边而不是上面或下面？
- 对于双面排版的文档，怎样确保一幅图形放置于奇数页或偶数页？还有，怎样确保两幅图形都出现在迎面的页上？
- 怎样得到带框的图形？
- 怎样得到并列的图形和子图？
- 怎样得到可跨页的连续图形？

如何得到本书？

本书的 PostScript 格式 (`epslatex.ps`) 和 PDF 格式 (`epslatex.pdf`) 的英文原版可从任一 CTAN (Comprehensive TeX Archive Network) 站点或其映像站点中获得。

England	`ftp://ftp.tex.ac.uk/tex-archive/info/`
Deutschland	`ftp://ftp.dante.de/tex-archive/info/`
Eastern U.S.	`ftp://tug2.cs.umb.edu/tex-archive/info/`
Western U.S.	`ftp://ftp.cdrom.com/pub/tex/ctan/info/`
Australia	`ftp://unsw.edu.au/tex-archive/info/`
Japan	`ftp://ftp.riken.go.jp/pub/tex-archive/info/`

完整的 CTAN 站点列表在任一 CTAN 站点或其映像站点中的 `CTAN.sites` 文件中，也可通过 fingering `ctan@ftp.dante.de` 来得到。中文版的 PDF 格式文件和源文件可从 ctex.dhs.org 下载。

王 磊

二〇〇〇年四月十三日

目录

一　背景知识

1 简介 3

2 LaTeX术语 5

3 Encapsulated PostScript 7

§3.1 禁止使用的 PostScript 操作符 7

§3.2 The EPS BoundingBox 8

§3.3 将 PS 转换为 EPS 9

§3.4 修正非标准的 EPS 10

4 怎样在LaTeX中使用 EPS 图 11

§4.1 行缓冲区溢出 12

5 下载和安装GhostScript 15

6 图像格式转换工具 17

§6.1 Level 2 EPS 封装 18

§6.2 编辑PostScript 19

二 LaTeX图形宏包

7 加入 EPS 图像文件 23

§7.1 includegraphics 命令 23

8 旋转和缩放对象 29

§8.1 scalebox 命令 30

§8.2 resizebox 命令 30

§8.3 rotatebox 命令 31

9 高级命令 33

§9.1 DeclareGraphicsExtensions 命令 34

§9.2 DeclareGraphicsRule 命令 35

三 LaTeX图形命令的使用

10 水平间距和居中 39

§10.1 水平居中 39

§10.2 水平间距 40

11 旋转、缩放和对齐 43

§11.1 高度和总体高度的区别 43

§11.2 旋转图形的放大和缩小 44

§11.3 旋转图形的对齐 45

§11.4 小页环境的垂直对齐 47

11.4.1 小页的底部对齐 48

11.4.2 小页的顶部对齐 49

12 使用子目录 51

§12.1 TEX搜索路径 52

§12.2 图形文件搜索路径 53

§12.3 节约 Pool 空间 53

13 压缩图形文件和非 EPS 文件的使用 57

§13.1 压缩 EPS 文件的例子 58

§13.2 TEX搜索路径和 dvips 59

§13.3 非 EPS 图形文件 60

13.3.1 GIF 的例子 61

13.3.2 对非 EPS 图形的直接支持 62

14 Psfrag 宏包 65

§14.1 Psfrag 使用例一 67

§14.2 Psfrag 使用例二 67

§14.3 EPS 图形中的 LATEX文本 69

§14.4 图形和文本的缩放 70

§14.5 Psfrag 的不兼容性 71

14.5.1 Xfig EPS 文件 71

§14.6 Overpic 宏包 71

15 多次使用同一图形的几种技巧 75

§15.1 定义PostScript命令 76

§15.2 在页眉和页脚使用图形 79

§15.3 在背景中使用图形水印 81

四 LaTeX图形环境

16 浮动图形环境 87
§16.1 创建浮动图形 88
§16.2 图形的放置 89
§16.3 清除未处理的浮动图形 91
§16.4 过多未处理的浮动对象 93
17 定制浮动位置 95
§17.1 浮动图形放置的计数器 95
§17.2 图形环境中的各种比例参数 96
§17.3 限制浮动 98
18 定制图形环境 99
§18.1 图形的间距 99
§18.2 图形上下方的水平线 100
§18.3 图形与标题的间距 102
§18.4 标题的标记 103
§18.5 将图形放于文档的最后 104
19 使用 caption2 宏包来定制标题 107
§19.1 标题式样 107
§19.2 标题式样的变换 108
§19.3 单行标题 109
§19.4 标题的宽度 111
§19.5 标题的分隔符 113
§19.6 标题的字体 114
§19.7 定制标题式样 116
§19.8 标题中的断行 118

§19.9 调整标题中的行距 119

20 不浮动的图形 121

§20.1 float 宏包中的 H 位置选项 122

21 边注图形 125

22 宽图形的处理 127

§22.1 单面版式中的宽图形 128

§22.2 双面版式中的宽图形 128

23 横排的图形 131

§23.1 Landscape 环境 132

§23.2 Sidewaysfigure 环境 134

§23.3 Rotcaption 命令 134

24 标题在一边的图形 137

§24.1 图形左侧标题 137

§24.2 图形内侧标题 138

§24.3 Sidecap 宏包 139

25 奇偶页中的图形 141

§25.1 迎面页图形 143

26 盒子中的图形 145

§26.1 图形在盒子中 145

§26.2 图形与标题均在盒子中 146

§26.3 定制 fbox 的参数 148

§26.4 Fancybox 宏包 148

27 并列的图形 151
§27.1 一图形环境中的并列图形 151
§27.2 并列的浮动图形 153
§27.3 并列的子图形 156
28 堆叠图形 161
29 图形与表格的平行排列 163
30 图文混排 165
§30.1 Wrapfig 宏包 165
§30.2 Picinpar 宏包 167
§30.3 Picins 宏包 169
31 连续图形 173
参考文献 175
索引 177

第一部分

背景知识

1

简 介

历史渊源

当 Knuth 编写 TEX 的时候，还没有 PostScript/EPS，JPEG，GIF 等图像格式，因此 DVI 并不直接支持这些格式的图形。不过，TEX允许 DVI 文件中包含 \special 命令来向 DVI 处理程序传递命令，这就使得 TEX 和 LATEX 能够使用 DVI 处理程序所支持的图像格式。

由于 DVI 文件经常被转为 PostScript 文件，所以支持最好的是对 EPS 格式 (Encapsulated PostScript, 是 PostScript 语言的子集) 的图像。在 LATEX 中插入 EPS 图像最初通过低层命令 \special 来完成。为方便起见，专门为 LATEX2.09 设计了两个高层的宏包 epsf 和 psfig。epsf 提供了 \epsfbox 命令来插入图片，另有三个命令来控制所插入的图片的缩放。而 psfig 中的 \psfig 命令除了用来插入图片，还可以缩小、放大、旋转它们。但是，尽管 psfig 的语法比较新颖，它的代码却没有 epsf 的健壮。于是作为这两个宏包的结合的产物，epsfig 宏包使用 psfig 的语法和大部分 epsf 的健壮代码。不过，epsfig 仍然使用了一些不健壮的 psfig 的代码。

LATEX 图形宏包套件

随着 1994 年 LATEX 2_ε 的发布，LATEX3 小组认识到在 LATEX 2_ε 中插入图形时的一些普遍问题，并且致力于开发出一个由全新命令组成的，比其它插图命令更加有效、更加健壮、更加方便的图形宏包套件 “LATEX graphics bundle[1]”。

LATEX 图形宏包套件包括 “标准” 的 graphics 宏包和 “扩展” 的 graphicx 宏包。这两个宏包都有一个 \includegraphics 命令，不过版本不同。 graphicx 版

[1]已经有 LATEX 图形宏包套件的 plain TEX 版本，可从 CTAN/macros/generic/graphics/ 得到相关的文件

的 \includegraphics 采用“命名机制”（类似 psfig 的语法），使用起来比较简单方便，却违犯了 LaTeX 可选参数的语法规则。作为一种妥协，就有了两种版本的 \includegraphics 。graphics 版本遵从 LaTeX 的语法规则，而 graphicx 版本则使用更为简便的命名机制。graphicx 版本的 \includegraphics 支持图形的缩放和旋转，而相应的 graphics 版本则要被置于 \scalebox 或 \rotatebox 才能达到同样的效果。

本书使用 graphicx 宏包是因为它比 graphics 宏包简便易用。尽管会使生成书中的图例的命令有些笨拙和缺少一点效率，这些例图同样可以用 graphics 宏包来完成。对这两个宏包详细的说明可参见 LaTeX 图形宏包套件的文档 [5]。

为保证对旧版本的兼容性，图形宏包套件中也提供了一个 epsfig 宏包，用以替代旧版本 epsfig。这一新版的 epsfig 中的命令如 \epsfbox，\psfig，\epsfig 只是做为 \includegraphics 的一个简单的封装，效率不高，只适合用来编译旧的文档。在编写新文档时要用 \includegraphics。

非 EPS 图形

LaTeX 图形宏包套件 还试图解决插入非 EPS 格式的图像如 GIF 和 JPEG 等的问题。由于 DVI 转换程序一般不支持直接插入大多数非 EPS 格式的图形，因此这些图形在加入到 LaTeX 文件前必须先转为 EPS 格式。在一些情况下，这一格式转换可由 DVI 到 PS 的转换程序自动完成。第 6 章介绍了一部分常用的图像格式转换工具，第 13 章则介绍了怎样在 LaTeX 中使用非 EPS 格式的图形。

2

LaTeX 术 语

任何 LaTeX 对象（字符，图形等）都把盒子作为单位 ([1, page 103])，每个盒子在它的左侧均有一参考点（*Reference point*）。盒子的基线（*baseline*，见图 2.1）是通过参考点的一条水平线。当 LaTeX 排列文本时，这些字符的参考点被从左到右的排成一条直线，称为当前基线（*current baseline*），并使它与字符的基线对齐。LaTeX 也用同样的方法来处理图形和其它对象，每个对象的参考点都被放置于当前基线上。

每个 LaTeX 盒子的大小由高度、深度、宽度（*height*,*depth*,*width*）来决定。高度是参考点到盒子顶部的距离，深度是参考点到盒子底部的距离，宽度则是盒

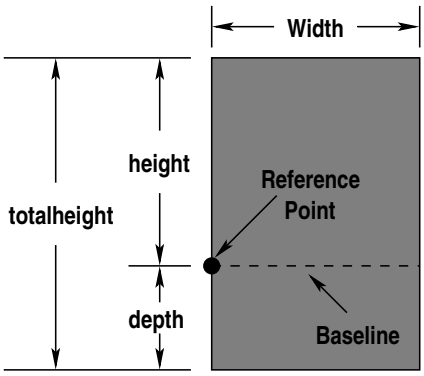


图 2.1: LaTeX 盒子示例

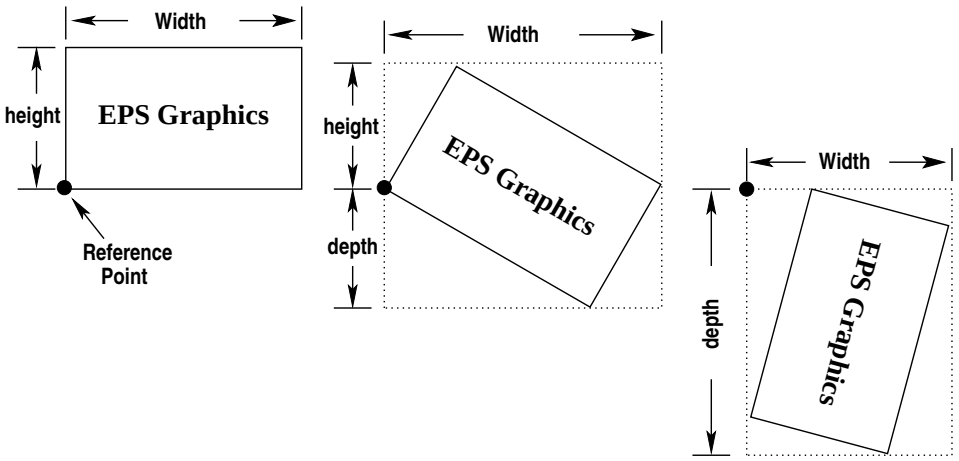


图 2.2: LaTeX 盒子的旋转示例

子的宽度。全部高度（*totalheight*）被定义为从盒子底部到顶部的距离，即：

$$\text{全部高度} = \text{高度} + \text{深度}$$

所有未曾旋转的 EPS 图形的参考点都是它的左下角（见图 2.2 的左边的盒子），它的深度为零，高度就等于全部高度。图 2.2 中间的盒子则是将图形旋转后，它的高度就不等于全部高度了。右边的盒子则展示可将图形旋转使其高度为零。

3

Encapsulated PostScript

PostScript 语言能够用来描述图形和文本。它既可在传统的 PostScript(PS) 文件中来描述多页的文档，也用于 Encapsulated PostScript(EPS) 文件中来描述插入文档的图形。PS 和 EPS 主要的区别在于：

- EPS 文件仅仅使用部分特定的 PostScript 操作符。
- EPS 文件必须含有一个 BoundingBox 行来确定 EPS 图形的大小。

§ 3.1. 禁止使用的 PostScript 操作符

由于 EPS 图形需要和其它对象一起共享页面，所以 EPS 文件中不能使用像选择页面大小 (`a4` 或 `letter`) 和清除整个页面 (`erasepage`) 等命令。下面是一些不能在 EPS 文件中使用的 PostScript 操作符：

`a3`	`a4`	`a5`	`banddevice`
`clear`	`cleardictstack`	`copypage`	`erasepage`
`exitserver`	`framedevice`	`grestoreall`	`initclip`
`initgraphics`	`initmatrix`	`letter`	`legal`
`note`	`prenderbands`	`quit`	`renderbands`
`setdevice`	`setglobal`	`setpagedevice`	`setpageparams`
`setsccbatch`	`setshared`	`startjob`	`stop`

尽管下列 PostScript 操作符可以在 EPS 文件中使用，但是不适当的使用它们极易导致错误。

```
nulldevice   setcolortransfer   setgstate    sethalftone
setmatrix    setscreen          settransfer  undefinedfont
```

上面的一些操作符可能会使 DVI 到 PS 的转换失败，另一些则可能导致像图形位置错误或图形消失等奇怪的问题。因为这些操作符绝大部分不会影响到 PostScript 的堆栈，所以，在大多数情况下，简单的将这些招致问题的操作符删除就可解决问题。其它的情形则需要更为复杂的 PostScript 的知识。

§ 3.2. The EPS BoundingBox

习惯上，PostScript 文件的第一行是标明该文件的类型，接下来的几行是被称为 *header* 或 preamble 的注释行（ PostScript 的注释符也是 % ）。这些注释中的一行就定义了 BoundingBox 。BoundingBox 这行有四个整数值，分别代表:

1. BoundingBox 的左下角的 x 坐标。
2. BoundingBox 的左下角的 y 坐标。
3. BoundingBox 的右上角的 x 坐标。
4. BoundingBox 的右上角的 y 坐标。

EPS 文件头示例

```
%!PS-Adobe-2.0 EPSF-2.0
%%Creator: gnuplot
%%DocumentFonts: Times-Roman
%%BoundingBox: 50 50 410 302
%%EndComments
```

上面的例子是一个由 `gnuplot` 生成的 EPS 文件的前五行。这个 EPS 图形的左下角的坐标是 $(50, 50)$，右上角的坐标是 $(410, 302)$。这里坐标的单位是 PostScript point，等于 $1/72$ 英寸。这样上面的这幅图的自然宽度为 5 英寸，相应

的自然高度为 3.5 英寸。需要注意的是 PostScript point 要比 TEX point（等于 1/72.27 英寸）稍大，在 TEX 和 LATEX 中，PostScript points 被称为 “big points” 或简称 `bp`，TEX point 被称为 “points” 或简称 `pt`。

§ 3.3. 将 PS 转换为 EPS

单页的 PostScript 文件，如果没有包含不适当的命令的话，可用下述方法转为 EPS 文件并加上 BoundingBox。**由于这些方法都不检查非法的 PostScript 操作符，所以只有在被转换的 PostScript 文件本身不含有那些被禁制使用的操作符的情况下，才能得到正确的 EPS 文件。**

1. 最方便的是用 GhostScript 里带的 `ps2epsi`（见第 5 章）。它可以读入 PostScript 文件并计算 BoundingBox 的参数，然后生成一个含有 PostScript 图形的 EPS 文件。

 最终得到的 EPS 文件是 EPSI 格式，即它在文件的开始部分带有一个底分辨率的预览位图。因为这个预览位图是 ASCII 编码的，所以不会造成像第 4.1 节的 `bufsize` 错误。不过，它却使得文件变大。

2. 另一种方法是计算 BoundingBox 的参数，然后把它加到 PostScript 文件中或作为插图命令的参数（比如用 `\includegraphics` 的 `bb` 方式）。计算 BoundingBox 的方法有以下几种:

 (a) 用 Ghostview 或 GSview 将 PostScript 图形打开，当鼠标在图形上移动时就会显示相应的坐标（以页面的左下角为参照点）。记下图形的左下角和右上角的坐标就可确定它的 BoundingBox。

 (b) 将 PostScript 图形打印一份，测量它的左下角和右上角到页面的左下角的水平和垂直距离（以英寸为单位），然后乘以 72 就可得到它的 BoundingBox。

 (c) 使用 `bbfig`。`bbfig` 是一个脚本文件，它在 PostScript 图形文件前面加入一些 PostScript 命令并送往 PostScript 打印机。这时加入的命令会计算 BoundingBox，然后将结果打印在 PostScript 图形上面。

§ 3.4. 修正非标准的 EPS

一些应用程序生成的非标准的 EPS 文件，另一些应用程序则根据它们自己的喜好来加入一些 PostScript 的“增强”功能，还有一些应用程序生成非常糟糕的 PostScript 代码。而由此得到的 EPS 文件是不能在 LaTeX 中使用的。所幸的是有许多有用的工具来修正这些非标准的 EPS 文件。

Mathematica 由 Mathematica 2.x 生成的 EPS 文件是用 Mathematica 的扩展 PostScript 写成的。在非 Mathematica 程序使用这些 EPS 文件时，必须要把那些非标准的扩展代码去掉才行。 DOS 版本的 Mathematica 2.x 带有一个名为 `printps.exe` 或 `rasterps` 的工具可以将那些非标准代码去掉。对于 Unix 版本的 Mathematica 2.x ，这个任务可由 `psfix` 来完成。参考你的 Mathematica 文档或与 Wolfram Research 联系来获取进一步的信息。

FrameMaker 由 FrameMaker 生成的 PostScript 文件没有遵循 Adobe 的与纸张无关的声明。 Framemaker 4 和 5 生成的 PostScript 文件可分别用下列脚本来修正：

ftp://ftp.irisa.fr/pub/FrameMaker/Filters/fixfm4-1.3.tar.gz

ftp://ftp.irisa.fr/pub/FrameMaker/Filters/fixfm5-2.0.tar.gz

修正 Framemaker 3 和 4 生成的 PostScript 文件的脚本 `fixfm3ps.sh` 和 `fixfm4ps.sh` 可从下面的地点得到：

ftp://ftp.frame.com/pub/techsup/framers/platform.ind/filters/

4

怎 样 在 LaTeX 中 使 用 EPS 图

EPS 文件能被 LaTeX 和 DVI 到 PS 的转换程序使用。

1. LaTeX 通过读取 EPS 文件中的 BoundingBox 行来决定为 EPS 图形保留多大的空间。
2. DVI 到 PS 的转换程序读取 EPS 文件并把它插入到生成的 PS 文件中。

需要说明的几种情形：

- 如果在图形插入命令中给定了 BoundingBox 的值， LaTeX 将不会从 EPS 文件读取它的 BoundingBox 行。
- 由于 TeX 不能读取非 ASCII 文件，也不能生成其它的程序，所以 LaTeX 不能从压缩的 EPS 文件或其它非 EPS 文件中得到 BoundingBox 的信息。在这种情况下，可以在图形插入命令中给定 BoundingBox 的值或将 BoundingBox 的值放到一个文本文件中（见第 13 章）
- EPS 图形并没有被加到 DVI 文件中，它是在从 DVI 到 PS 转换时才被加到生成的 PS 文件中的。因此，所有用到的 EPS 文件必须和 DVI 文件在一起。
- 大多数旧版本的 DVI 浏览器不支持显示 EPS 图形。这时， DVI 浏览器一般会将 EPS 图形的 BoundingBoX 用一方框显示出来，以方便使用者对图形

进行定位。目前版本的一些 TeX 软件如 MikTEX 、 fpTEX 和 teTEX 等所带 DVI 浏览器（`Yap`, `Windvi`, `Xdvi`）可以借助于 `ghostscript` 来显示 EPS 图形。

§ 4.1. 行缓冲区溢出

LATEX 在读取 ASCII 文件时是每次从中读取一行，然后把它放到自己的行缓冲区里。 LATEX 的行缓冲区大约有 3000 字节。如果 EPS 文件中有一行的长度超过了行缓冲区的长度，就会产生如下的错误讯息：

```
Unable to read an entire line--bufsize=3000.
Please ask a wizard to enlarge me.
```

因为 EPS 很少有一行长度超过 3000 字节的情形，所以产生行缓冲区溢出的原因可能有两种：

1. EPS **文件中有一个长的二进制的预览图**

 有些应用程序生成的 EPS 文件在开始部分放置了一个二进制的预览图，这样就可使得像 DVI 浏览器等一些不能解释 PostScript 的软件也可来显示 EPS 图形。目前有少数与 TEX 有关的软件使用这种方法。

 如果这个二进制的预览图比行缓冲区小， `\includegraphics` 将会略过它（像 `\psfig` 等过时的命令则不会这样）。但是，如果这个二进制的预览图比行缓冲区大的话，就会发生行缓冲区溢出的错误。有两种解决办法：

 (a) 如果不需要预览图，可以用文本编辑器将它删掉或在生成 EPS 图形时就选择不要预览图。

 (b) 因为 LATEX 读取 EPS 文件的唯一目的就是取得 BoundingBox 的大小，故可在插图命令中给出 BoundingBox 的值（如在 `\includegraphics` 中使用 `bb` 选项）从而使得 LATEX 不再读取 EPS 文件。

2. EPS **文件中的分行符在不适当的传输中被损坏**

 （这里所谈到的问题不会在一些最新版本的 TEX 软件中出现，因为这些软件中的 TEX 都会正确的识别所有的分行符。）

不同的操作系统平台使用不同分行符。Unix 使用 `^J`，Macintosh 使用 `^M`，而 DOS/Windows 则使用 `^M^J`。比如一个 EPS 文件从 Macintosh 机上用二进制方式传输到 Unix 机上，那么 Unix 机上的 TeX 会因找不到分行符 `^J` 而把整个文件作为一行，导致行缓冲区溢出的错误。

如果 EPS 文件中不含有二进制的部分（如预览图和嵌入的图形），以文本方式传输就可以解决这一问题。否则，由于文件必须用二进制方式传输，分行符的问题不可避免，这时就需要用一些工具来转换不同的格式或在在插图命令中给出 BoundingBox 的值来解决。

5

下 载 和 安 装 GhostScript

GhostScript 是一个 PostScript 语言解释器，它可以运行在大多数操作系统平台上并由 Aladdin Enterprises 自由[1]发放。通过 GhostScript 可以在屏幕上显示 PostScript 和 EPS 文件，也可用非 PostScript 打印机来打印。 Aladdin GhostScript 可从 CTAN/supported/ghostscript/aladdin/ 取得。也可直接访问 GhostScript 的主页：

http://www.cs.wisc.edu/ ghost/index.html

这一 Web 网址提供了比 CTAN FTP 站点更多更好的相关信息。这些站点都提供 Windows/DOS/OS/2 和 Macintosh 的可执行文件， Unix/VMS 下的源代码。同时，还能下载 GhostScript 的图形用户界面（ Windows/OS/2 下的 GSview 和 Unix/VMS 下的 Ghostview ），这将使你更容易的观看和打印 PostScript 文件。

Aladdin GhostScript 目前的正式版本为 6.01， GNU GhostScript 的正式版本为 5.50。对于 Window/DOS/OS/2 和 Macintosh 的用户来说，直接下载它的已编译好的压缩包，安装使用就行了。而对于 Unix/VMS 的用户，通常需要自己编译。编译时，首先将 `ghostscrip-x.xx.tar.gz` 和其它所需的软件包解压缩：

[1] Although Aladdin Ghostscript is distributed for free, it is not in the public domain. It is copyrighted and comes with certain limitations such as no commercial distribution. When versions of Aladdin Ghostscript become approximately one year old, Aladdin releases them as “GNU Ghostscript” whose use is governed by the less-restrictive GNU Public License.

```
gzip -dc ghostscrip-x.xx.tar.gz | tar -vxf -
cd gsx.xx
gzip -dc ghostscrip-x.xxjpeg.tar.gz | tar -vxf -
gzip -dc ghostscrip-x.xxlibpng.tar.gz |tar -vxf -
gzip -dc ghostscrip-x.xxzlib.tar.gz | tar -vxf -
```

解压后可参考 GS 所带的帮助文件 make.txt 来按照你的要求编辑适当的 .mak 文件，然后进行编译（假设使用 gcc 编译）和安装：

```
ln -s src/gcc.mak ./Makefile
make
make install
```

GhostScript 中还带有一些有用的工具，如 ps2pdf 等，可利用 GS 来转换图形，打印、预览 PostScript 文件。详细的使用说明可参考 GS 所带的使用说明文件。

6

图 像 格 式 转 换 工 具

下面列出的一些免费软件和共享软件可以用来将非 EPS 格式的图形转换为 EPS 图形。其中一部分提供命令行方式的软件，在用 dvips 将 DVI 转为 PS 的过程中能同时自动转换图形的格式。具体见第 13.3 节。

- ImageMagick 是一个很好的图形转换工具，可从 ftp.wizards.dupont.com 或其它站点自由下载。参见：

 http://www.wizards.dupont.com/cristy/ImageMagick.html

 除 Unix 和 Linux 外，它也可运行在 Windows NT, Macintosh 和 VMS 下。

- xv 是一个 $25 的共享软件，运行在 X-Windows 环境下，可用来观看和转换图形。 xv 没有命令行方式，因此无法利用她来实现图形格式的即时转换功能。有关 xv 的信息可参见：

 http://www.sun.com/sunsoft/catlink/xv/note.html

 xv 的在线手册：

 http://is.rice.edu/ shel/xv-3.10a/

- DISPLAY 是 DOS 下的免费软件，能够转换多种图像格式。可从下面的地点下载 disp189a.zip 和 disp189b.zip （新的版本可能不是 189 。）

 http://www.simtel.net/simtel.net/msdos/graphics-pre.html

 http://www.simtel.net/simtel.net/msdos/graphics-pre.html

- `WMF2EPS` 运行在 Windows9.x/NT 下的将 WMF 格式的图像转为 EPS 格式的免费软件。参考以下文件来取得这一软件。

 `CTAN/support/wmf2eps/readme.txt`

 它需要你的系统中装有 Adobe 兼容的打印机驱动。

- `KVEC` 是一个 $25 的共享软件，能够将位图格式的图形（BMP, GIF, TIFF）等转为 PostScript 或其它矢量图形。`KVEC` 可运行在 Windows, OS/2, NEXT 和 Unix 下。

 `http://ourworld.compuserve.com/homepages/kkuhl/`

- `NetPBM` 是旧有的 `PBMPLUS` 工具包的保留和扩充。目前它可运行在 Windows, Unix, VMS, DOS 等多种平台下。

 `http://wuarchive.wustl.edu/graphics/graphics/packages/NetPBM/`

- `ImageCommander`（共享软件，$19）是 Windows3.1/95/NT 下的图形转换软件，可将 GIF, JPEG, PICT, WMF 等多种图形转换为 EPS 和其它格式的图形。更详细的信息可见：

 `http://www.jasc.com/`

 JASC 的 Paint Shop Pro 绘图软件（共享软件，$69）具有同样的图形转换功能。

§ 6.1. Level 2 EPS 封装

与传统的 PostScript 不同的是，Level 2 PostScript 支持压缩的二进制图形。这使得它能够制作出比传统的 EPS 更小、质量更好的图形。如果你有一台 Level 2 PostScript 打印机，那么你最好是用下面的这些封装程序来替代上节中的那些转换软件。不过由于这样得到的 PostScript 文件只能在 Level 2 PostScript 打印机上打印，会降低文件的通用性。

- `jpeg2ps` 是一个用 C 语言程序，可将 JPEG 图形转换为 Level 2 PostScript 图形。`jpeg2ps` 可在 Unix, DOS 和其它操作系统下使用。

 `http://www.muc.de/ tm/free/free.html`

- TIFF 图形可用 `tiff2ps` 转换为 LZW–编码的 Level 2 PostScript 。`tiff2ps` 的源代码在：

 ftp://ftp.sgi.com/graphics/tiff/tiff-v3.4-tar.gz

 `tiff2ps` 能在 Unix, DOS, Mac 和 VMS 下编译成功。尽管 LZW PostScript 文件较小，但是它需要 Level 2 PostScript 打印机。

§ 6.2. 编辑 PostScript

虽然可直接编辑 EPS 文件中的 PostScript 命令来改变图形，但这对一些不熟悉 PostScript 语言的人来说还是很困难的。所幸的是，借助于下面的一些工具软件，可以很容易的编辑 EPS 图形。

- `pstoedit` 是 Unix, Windows, DOS 和 OS/2 下的免费软件，借助于 GhostScript，它能购将 PostScript 或 PDF 图形转为其它矢量格式（比如 `xfig` 的 `.fig` 格式）。`pstoedit` 的C++ 源码可从下述站点取得。

 ftp://ftp.x.org/contrib/applications/pstoedit/pstoedit.html
 http://www.geocities.com/SiliconValley/Network/1958/pstoedit/

- `Mayura Draw`（以前称为 `PageDraw`）是 Windows3.1/9.x/NT 下的绘图软件。当与 GhostScript 一起使用时，它可以编辑 PostScript 文件。见：

 http://www.wix.com/PageDraw

 旧版本的 `Mayura Draw` 是免费软件，最近的版本则为 $15 的共享软件。`Mayura Draw` 需要 Adobe Type Manager (ATM) 来在图形上放置文本。虽然 ATM 现在是商业软件，但是 Adobe 在 Acrobat Reader 2.0 中提供了一个免费版本。

 ftp://ftp.winsite.com/pub/pc/win3/util/acroread.zip

- `xfig` 是 Unix/Xwindows 下功能强大的免费绘图软件，能够引入 EPS 图形并加上标记。不过目前还不能改变原始的 EPS 图形。访问 `xfig` 的主页获取更进一步的信息。

 http://www.xfig.org/

第二部分

图形宏包

7

加 入 EPS 图 像 文 件

关于 graphics 和 graphicx 宏包的最好的参考资料是 *Graphics guide*[5] 和 *LaTeX Graphics Companion*[4]。其它的 LaTeX 参考资料中对 graphicx 包只是零星的介绍。[2] 中对 graphics 和 graphicx 都作了介绍，[1] 中只介绍了 graphics包，而 [3] 中两者均未提及。

§ 7.1. includegraphics 命令

```
\includegraphics[选项]{文件}
```

这里的选项在表 7.1, 7.2, 7.3 中列出。因为 \includegraphics 不会结束当前段落，所以它能够在文本中放置图形如  和 。下面的命令将以 file.eps 的自然大小插入到 LaTeX 文档中：

```
\documentclass{article}
\usepackage{graphicx}
\begin{document}
\includegraphics{file.eps}
\end{document}
```

如果加入的图形文件没有指明扩展名，那么 \includegraphics 会根据 \DeclareGraphicsExtensions 的扩展名列表自动为它加上扩展名（见第 9.1

表 7.1: includegraphics Options

height	图形的高度（可为任何 TEX 度量单位）。
totalheight	图形的全部高度，可为任何 TEX 度量单位（*6/95 增加*）。
width	图形的宽度（可为任何 TEX 度量单位）。
scale	图形的缩放因子，设定 scale=2 会使插入的图形的大小为其自然大小的两倍。
angle	设定旋转的角度，以度为单位，顺时钟方向为正。
origin	origin 指定图形绕那一点旋转，缺省是图形的参考点（*12/95 增加*）。初始点有可能与第 8.3节的 \rotatebox 命令中的一样。比如 origin=c 将使图形绕它的中心旋转。
bb	设定 BoundingBox 的值。bb=10 20 100 200 设定 BoundingBox 的左下角在 (10,20)，右上角在 (100,200)。因为 \includegraphics 会自动从 EPS 文件中读入 BoundingBox 行所给的值，所以一般不使用 bb 这个选项。但它在 EPS 文件中的 BoundingBox 丢失或出错时还是很有用的。

表 7.2: includegraphics Cropping Options

viewpoint	指定图形可以被看到的部分。如同 BoundingBox 一样，这是一个由四个数字，左下角和右上角的坐标所确定的区域。这里的坐标是相对于 BoundingBox 的左下角的（*6/95 增加*）。 例如，如果图形的 BoundingBox 的值是 50 50 410 302，viewpoint=50 50 122 122 将显示以图形的左下角为左下角的一英寸大小的区域。而 viewpoint=338 230 410 302 则会显示以图形的右上角为右上角的一英寸大小的区域。 必须使用 clip 选项（见表 7.3）来阻止显示视图以外的图形部分。
trim	指定图形可以被看到的部分的另一选项。所给出的四个数字分别代表了从左、下、右、上被截去的值。正数代表从此方向截去的大小，而负数则代表从此方向加上的大小。

表 7.3: includegraphics Boolean Options

noclip	（缺省选项）显示整个的图形，即使有些部分在视图之外。
clip	当使用 clip 时，将不显示图形在视图之外的部分。
draft	当使用 draft 选项时，将只显示图形的 BoundingBox 和文件名，这使得显示和打印文档的速度加快。如果使用 draft 宏包选项，\usepackage[draft]{graphicx} 会导致文档中的所有图形都被以草稿（draft）方式插入。
final	（缺省选项，除非使用\usepackage[draft]{graphicx}）final 选项使得图形被显示，经常用来覆盖 \usepackage[draft]{graphicx}
keepaspectratio	在没有设定 keepaspectratio 选项时，给定图形的高度（全部高度）和宽度会导致图形被不对称缩放来满足所设定的高和宽。在设定 keepaspectratio 选项后，给定图形的高度（全部高度）和宽度时，图形会保持原有的宽高比例，尽可能使得图形满足所设定的高和宽，但是图形不会超出其中任一个。

节）。由于缺省的扩展名列表不包括空的扩展名，\includegraphics{file} 不会读入 file。除非空的扩展名已被加到扩展名列表中。

命令 **指定宽度**

```
\includegraphics[width=3in]{file.eps}
```

将 file.eps 插入文档并且它的宽度被缩放到 3 英寸，高度也会按相应的比例缩放。如果用 \textwidth 或 \em 等的函数来指定宽度，而不是用像 3 英寸这样的固定尺寸，将会使你的 LaTeX 文档更具通用性。例如：

```
\includegraphics[width=\textwidth]{graphics.eps}
```

将所插入图形缩放到和文本行的宽度一样宽。而下面的命令

```
\includegraphics[width=0.80\textwidth]{graphics.eps}
```

使得插入图形的宽度为文本行宽的 80%。当与 calc 宏包配合使用时，下面的命令可令图形的宽度比文本行宽少 2 英寸：

```
\includegraphics[width=\textwidth-2.0in]{graphics.eps}
```

（需要 graphicx *12/95* 或以后的版本。）

下面是一些使用 \includegraphics 命令来插入图形的例子。这里为方便起见，定义 \HR 如下：

```
\newcommand{\HR}{\rule{1em}{0.4pt}}
```

在下面的几个例子中，可以比较以下使用 scale,width,height,angle 以及 keepaspectratio 选项及其不同的顺序所得到的不同效果。



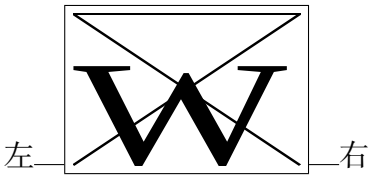
例一

```
左\HR\fbox{%
  \includegraphics
     [scale=.5]{w.eps}%
  \HR 右
```



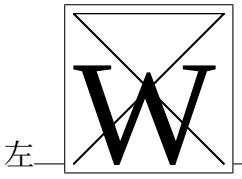
例二

```
左\HR\fbox{%
  \includegraphics%
     [width=10mm]{w.eps}%
  \HR 右
```



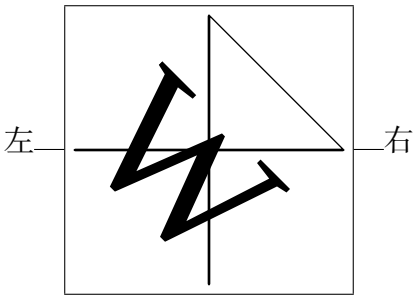
例三

```
左\HR\fbox{%
  \includegraphics
     [height=20mm,width=30mm]%
     {w.eps}}\HR 右
```



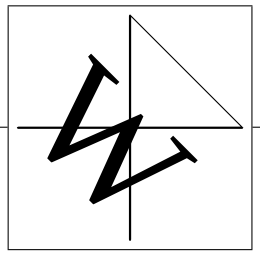
例四

```
左\HR\fbox{%
  \includegraphics
     [height=20mm,width=30mm,%
      keepaspectratio]{w.eps}}%
  \HR 右
```



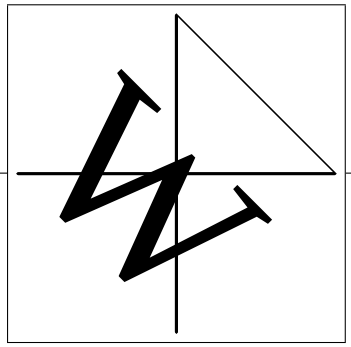
例五

```
左\HR\fbox{%
  \includegraphics
     [angle=-45]{w.eps}}%
  \HR 右
```

左右

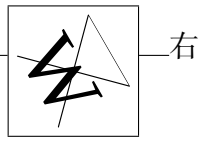
例六

```
左\HR\fbox{%
  \includegraphics
     [angle=-45,width=30mm]%
     {w.eps}}\HR 右
```

左右

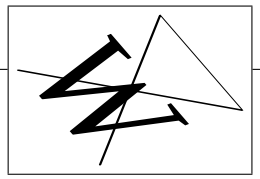
例七

```
左\HR\fbox{%
  \includegraphics
     [width=30mm,angle=-45]%
     {w.eps}}\HR 右
```

左右

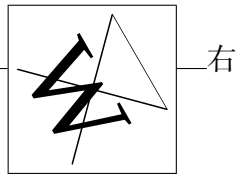
例八

```
左\HR\fbox{%
  \includegraphics
     [angle=-60,totalheight=15mm]%
     {w.eps}}%
\HR 右
```

左右

例九

```
左\HR\fbox{%
  \includegraphics
     [angle=-60,totalheight=20mm,%
      width=30mm]{w.eps}}%
\HR 右
```

左右

例十

```
左\HR\fbox{%
  \includegraphics
     [angle=-60,totalheight=20mm,%
      width=30mm,keepaspectratio]%
     {w.eps}}%
\HR 右
```

8

旋 转 和 缩 放 对 象

除了上一章介绍的 \includegraphics 命令外， graphicx 宏包还提供了另外四个命令用来旋转和缩放任意的 LaTeX 对象：文本， EPS 图形等等。

```
\scalebox{水平缩放因子}[垂直缩放因子]{对象}
\resizebox{宽度}{高度}{对象}
\resizebox*{宽度}{全部高度}{对象}
\rotatebox[选项]{角度}{对象}
```

因为 graphicx 包的 \includegraphics 带有支持旋转和缩放的 angle 和 width 等选项，所以本章介绍的这几个命令很少在插图时使用。例如：

```
\includegraphics[scale=2]{file.eps}
\includegraphics[width=4in]{file.eps}
\includegraphics[angle=45]{file.eps}
```

上述命令和下面的命令等到的结果是相同的。

```
\scalebox{2}{\includegraphics{file.eps}}
\resizebox{4in}{!}{\includegraphics{file.eps}}
\rotatebox{45}{\includegraphics{file.eps}}
```

尽管结果相同，但在实际使用中最好还是用前一种方法，因为它能更迅速的生成效率更高的 PostScript。

§ 8.1. scalebox 命令

```
\scalebox{水平缩放因子}[垂直缩放因子]{对象}
```

\scalebox 命令对其作用的对象进行缩放，使缩放后的对象的宽度为原始宽度与水平缩放因子之积，高度为原始高度与垂直缩放因子之积。如果垂直缩放因子没有给出，那么将按照给定的水平缩放因子，保持原始宽高的比例进行缩放。如果缩放因子为负值，则对对象进行反射。下面是几个例子：

这是放大的文字
这是正常的文字
这是缩小的文字

```
\scalebox{2}{这是放大的文字} \\
          这是正常的文字  \\
\scalebox{.5}{这是缩小的文字}
```

放大
和
缩小

放大
和
缩小

```
\framebox{\scalebox{2}{%
  \parbox{.5in}{放大 \\ 和 \\ 缩小}}}
\framebox{\scalebox{2}[1.5]{%
  \parbox{.5in}{放大 \\ 和 \\ 缩小}}}
```

China? China?
China? China?
China? China?

```
China? \scalebox{-1}[1]{China?} \\
China? \scalebox{1}[-1]{China?} \\
China? \scalebox{-1}[-1]{China?} \\
China? \scalebox{-1}{China?}}
```

§ 8.2. resizebox 命令

```
\resizebox{宽度}{高度}{对象}
\resizebox*{宽度}{全部高度}{对象}
```

\resizebox 命令将对象的大小改变为给定值。如果宽度 或高度中的任一项用 ! 给出，则其代表的选项的长度在被改变大小时会保持原有的宽高比例不变。

例如：`\resizebox{2in}{!}{argument}` 将对象的宽度改变为 2 英寸。

标准的 LaTeX 2_ε 长度 `\height`, `\width`, `\totalheight`, `\depth` 可用来表示对象的原始尺寸。因此，`\resizebox{2in}{\height}{argument}` 使得对象的宽度改变为 2 英寸但保持原有高度不变。除了第二个参数表示对象的全部高度以外，`\reseizebox*` 与 `\resizebox` 是相同的。下面是几个例子：

选项
角度
对象

```
\framebox{\resizebox{5mm}{!}{%
   \parbox{14mm}{选项 \\ 角度 \\ 对象}}}
\framebox{\resizebox{!}{10mm}{%
   \parbox{14mm}{选项 \\ 角度 \\ 对象}}}
```




```
\resizebox*{2cm}{3cm}{\LaTeX{}~图形} \\
\resizebox*{2cm}{1cm}{\LaTeX{}~图形}
```

§ 8.3. rotatebox 命令

```
\rotatebox[选项]{角度}{对象}
```

`\rotatebox` 将对象旋转一给定度数的角度，逆时针方向为正。缺省地，对象绕它的参考点旋转。`\rotatebox` 命令中的 选项 允许对象绕给定的点来旋转。

1. 给定 `[x=xdim,y=ydim]`，则对象旋转所绕的点相对于参考点的坐标为 `(xdim,ydim)`。

2. `origin` 选项指定 12 个特殊点其中之一（见图 8.1）。

 `origin` 点的水平位置由 `lcr`（分别代表左、中、右）其中之一确定，垂直位置则由 `t,c,B,b`（分别代表顶部、中部、基线、底部）中的一个来确定。例如：

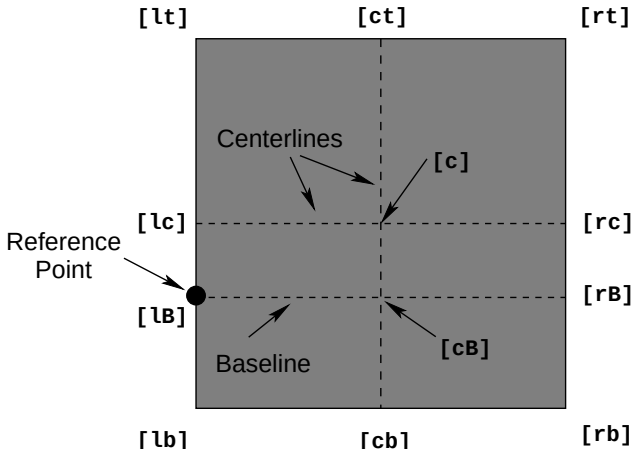


图 8.1: Available Origin Point

`[rb]` 右下角。

`[lt]` 左上角。

`[cB]` 图形基线的中点。

几点说明：

- 标记字母的顺序并不重要，`[br]` 就等于 `[rb]`。
- `c` 代表水平位置的中点还是垂直位置的中点靠和它一起的标记字母来决定。
- 如果只给出一个标记字母，那么另一个将被假设为 `c`。即 `[c]` 等于 `[cc]`，`[l]` 等于 `[lc]`，`[t]` 等于 `[ct]` 等等。

下面是一个例子：

旋转 0° 旋转 45° 旋转 90° 旋转 135° 旋转 180° 旋转 225°

```
\setlength{\fboxsep}{0mm}
\newcommand{\MyRot}[1]{%
   \fbox{\rotatebox{#1}{旋转~$#1^\circ$}}}
\MyRot{0} \MyRot{45} \MyRot{90}
\MyRot{135} \MyRot{180} \MyRot{225}
```

9

高 级 命 令

本章描述了一些在下述情形下使用的 LaTeX 2$_\varepsilon$ 图形宏包套件的高级命令。

1. 当使用没有扩展名的文件名时。如：

   ```
   \includegraphics{file}
   ```

2. 当使用压缩的 EPS 图形文件时。见第 13.1 节。

3. 当使用非 EPS 格式的图形文件时。见第 13.3 节。

在这些情况下， LaTeX 如何处理由 `\includegraphics` 所引入的文件，就需要用 `\DeclareGraphicsRule` 和 `\DeclareGraphicsExtensions` 命令来控制。

- `\DeclareGraphicsExtensions` 命令指定了在没有提供图形文件扩展名的情况下， LaTeX 将自动为其加上的扩展名列表（如 `.eps`，`.ps`，`.eps.gz` 等）。

- `\DeclareGraphicsRule` 命令指定了对图形文件执行的命令。执行这一命令要求操作系统支持管道功能，比如 Unix 等操作系统，而 DOS 则不行。

 若将此命令指定为一解压缩命令，那么就可以使用压缩的 EPS 图形文件。若将此命令指定为一图形格式转换命令，那么就可以使用非 EPS 格式的图形文件。

§ 9.1. DeclareGraphicsExtensions 命令

\DeclareGraphicsExtensions 命令告诉 LaTeX，若 \includegraphics 命令所引入的文件没有提供扩展名，将试图为其自动加上什么样的扩展名。为方便起见，在选择图形驱动[1]时，就已经有一个相应的预设的扩展名集。举例来说，如果选择 dvips 作为图形驱动，那么缺省地会使用下列图形文件扩展名（在 dvips 中定义）：

```
\DeclareGraphicsExtensions{.eps,.ps,.eps.gz,.ps.gz,.eps.Z}
```

这时，\includegraphics{file}让 LaTeX 首先寻找 file.eps，其次 file.ps，再其次 file.eps.gz，直到找到一个文件。相应地，你就可以在 LaTeX 文件中用

```
\includegrapincs{file}
```

取代

```
\includegrapincs{file.eps}
```

这样做的好处是如果你以后决定压缩 file.eps，你也无须更改 LaTeX 文件。

无扩展名的文件

说明：

```
\includegrapincs{file}
```

不会试图寻找 file，除非空的扩展名 {} 已被加入到扩展名列表中。例如：

```
\DeclareGraphicsExtensions{.eps,.eps.gz,{}}
```

将试图在没找到 file.eps 和 file.eps.gz 的情况下寻找 file。

Pool Space 问题

不给出扩展名 而靠 LaTeX 从 \DeclareGraphicsExtensions 的扩展名列表中选择正确的扩展名可能加重 pool space 问题（见第 12.3 节）。如果有 pool space 问题的话，应当使扩展名列表中的扩展名数目尽可能小。如：

```
\DeclareGraphicsExtensions{.eps,.eps.gz}
```

[1]指定一个图形驱动选项如 \usepackage[dvips]{graphics} 将会覆盖掉在 graphics.cfg 中设定的缺省驱动选项

表 9.1: DeclareGraphicsRule Arguments

ext	文件的扩展名。
type	扩展名所对应的图形格式。
sizefile	包含图形的 BoundingBox 的文件的扩展名。如果这一选项为空，那么须要在 \includegraphics 命令中给定 bb 项的值。
command	作用于图形文件的命令，此项常为空。命令前必须有一个后向单引号（而不是常使用的前向单引号）。目前为止，只有 dvips 能够使用这样的命令。参见第 13 章用这样的命令来处理非 EPS 格式图形和压缩 EPS 图形的例子。

§ 9.2. DeclareGraphicsRule 命令

\DeclareGraphicsRule 命令指定 \includegraphics 如何按照文件的扩展名来对图形文件进行操作。可以允许有多个 \DeclareGraphicsRule 命令。

```
\DeclareGraphicsRule{ext}{type}{sizefile}{command}
```

例如：

```
\DeclareGraphicsRule{.eps.gz}{eps}{.eps.bb}{`gunzip -c #1}
```

指定任何以 .eps.gz 为扩展名的文件为压缩 EPS 文件，该文件的 BoundingBox 信息存放在扩展名为 .eps.bb 的文件中，并用命令 gunzip -c 来解压缩（因为 LaTeX 不能从压缩文件中读取 BoundingBox 信息，所以 BoundingBox 行必须存放到一非压缩文件中）。

\DeclareGraphicsRule 命令允许使用 * 代表任何未知扩展名，例如：

```
\DeclareGraphicsRule{*}{eps}{*}{}
```

会导致所有未知扩展名的文件都被认为是 EPS 文件，比方说 file.EPS 就被当做 EPS 文件。

文件名中的句点

这里文件名里第一个句点 以后的部分都被认为是文件的扩展名，这样做是为了能够正确地识别压缩的 EPS 文件（扩展名为 .eps.gz）等。为了避免混淆，文件的基本名中不要使用句点。否则 file.name.eps.gz 会让 \includegraphics 寻找扩展名为 .name.eps.gz 所对应的规则，由于这样的规

则很有可能不存在，结果导致使用未知扩展名所对应的规则。例外的情形是该文件的格式正好是缺省格式，如未知扩展名的文件都被认为是 EPS 文件时，那么 `file.name.eps` 就能被正确地识别。

预先定义的命令

为方便起见，根据不同的图形驱动选项[2] 预定义了不同的缺省图形规则。例如使用 `dvips` 图形驱动选项时，缺省图形规则为：

```
\DeclareGraphicsRule{.eps}{eps}{.eps}{}
\DeclareGraphicsRule{.ps}{eps}{.ps}{}
\DeclareGraphicsRule{.pz}{eps}{.bb}{`gunzip -c #1}
\DeclareGraphicsRule{.eps.Z}{eps}{.eps.bb}{`gunzip -c #1}
\DeclareGraphicsRule{.ps.Z}{eps}{.ps.bb}{`gunzip -c #1}
\DeclareGraphicsRule{.eps.gz}{eps}{.eps.bb}{`gunzip -c #1}
\DeclareGraphicsRule{.ps.gz}{eps}{.ps.bb}{`gunzip -c #1}
\DeclareGraphicsRule{.pcx}{bmp}{}{}
\DeclareGraphicsRule{.bmp}{bmp}{}{}
\DeclareGraphicsRule{.msp}{bmp}{}{}
\DeclareGraphicsRule{*}{eps}{*}{}
```

前面两个命令定义扩展名为 `.eps` 和 `.ps` 的文件为 EPS 文件，它们后面的五个命令定义了压缩 EPS 文件的扩展名和解压命令，接下来的三个命令定义了位图文件的扩展名（见第 13.3 节），最后一个命令设定未知扩展名的文件为 EPS 文件。

[2]指定一个图形驱动选项如 `\usepackage[dvips]{graphics}` 将会覆盖掉在 `graphics.cfg` 中设定的缺省驱动选项

第三部分

图形命令的使用

10

水 平 间 距 和 居 中

§ 10.1. 水平居中

图形的放置位置由当前文本的排列方式所决定。为使图形居中放置，可将其放入一个居中（center）环境中。

```
\begin{center}
   \includegraphics[width=2in]{graphic.eps}
\end{center}
```

如果将 \includegraphics 命令放入一个环境中（minipage 或 figure），用 \centering 可将其后的内容居中排列。例如：

```
\begin{figure}
   \centering
   \includegraphics[width=2in]{graphic.eps}
\end{figure}
```

就等同于

```
\begin{figure}
\begin{center}
   \includegraphics[width=2in]{graphic.eps}
\end{center}
\end{figure}
```

这里推荐使用 \centering，因为 \begin{center} 会使图形上下方的垂直间距增加一倍（ figure 环境带有的间距加上 center 环境带有的间距 ）。若希望有特殊的垂直间距，可使用第 18.1 节介绍的命令。

过时的用法

\psfig 和 \epsfbox 命令 的缺陷让它们很难使图形居中排列。作为一种解决办法，使用了 TEX 命令 \centerline 和 \leavevmode。而 \includegraphics 命令已克服了这些缺陷，允许直接与 \centering 命令一起使用或用在 center 环境中，因此也就不需再使用 \centerline 和 \leavevmode了。

§ 10.2. 水平间距

LATEX 在排列图形的时候实际上与排列其它的像文字这样的对象是一样的，了解到这一点很重要。举例来说，如果行尾不是以 % 结束的话， LATEX 会自动在两行之间加进一个字符的水平间距。像：

```
朋友
你好
```

在输出结果中“朋友” 和 “你好” 之间会有一个字符的水平间距。

```
\includegraphics{file.eps}
\includegraphics{file.eps}
```

则在图形之间有一个字符的水平间距。在第一行的行尾加上一个 %

```
\includegraphics{file.eps}%
\includegraphics{file.eps}
```

就会使图形之间没有水平间距。如果需要，可用 \hspace 命令在图形之间加进指定长度[1]或用 \hfill 来加进一个可填充可能的间距的橡皮长度。例如：

```
\includegraphics{file.eps}\hfill\includegraphics{file.eps}
```

将两个图形尽量向左右分开。而

[1]用 \textwidth 或 \em 等的函数作为 \hspace 的参数，而不是采用一固定度量，可提高文档的通用性。

```
\hfill\includegraphics{file.eps}%
\hfill\includegraphics{file.eps}\hspace*{\fill}
```

使得图形的两边和中间的间距都相等。由于换行符前的 \hfill 命令将被忽略，所以需要用 \hspace*{\fill} 来替代它。

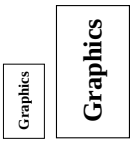
11

旋 转 、 缩 放 和 对 齐

因为 \includegraphics 的选项是从左到右依次处理的，所以角度和大小选项的顺序不同会导致不同的结果。如

```
\begin{center}
   \includegraphics[angle=90,totalheight=1cm]{graphic.eps}
   \includegraphics[totalheight=1cm,angle=90]{graphic.eps}
\end{center}
```

的输出结果为：



第一个命令使得图形被旋转 90 度后缩放为 1 厘米高，而第二个命令则先将图形缩放为 1 厘米高然后再旋转 90 度。

§ 11.1. 高度和总体高度的区别

在使用 height 选项时要特别小心，仅管它经常意味着由 totalheight 选项给出的全部高度（见第 5 页图 2.1）。在对象的深度为零时，对象的全部高度就是它的高度，使用 height 选项不会有什么问题。但是，当对象的深度不为零

时，使用 height 而不是 totalheight 会导致不正确的图形大小或除零的错误。对于外部的 EPS 图形，区分 height 和 totalheight 尤其在旋转和缩放时显得特别重要。例如：

```
\includegraphics[angle=-45,totalheight=1in]{file.eps}
\includegraphics[angle=-45,height=1in]{file.eps}
```

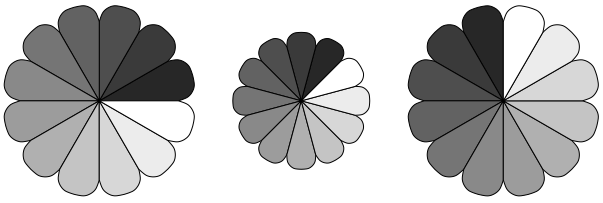
第一个命令缩放一个旋转了的图形，使其全部高度为 1 英寸。而第二个命令缩放一个旋转了的图形，使其在参考点以上的部分为 1 英寸高。

§ 11.2. 旋转图形的放大和缩小

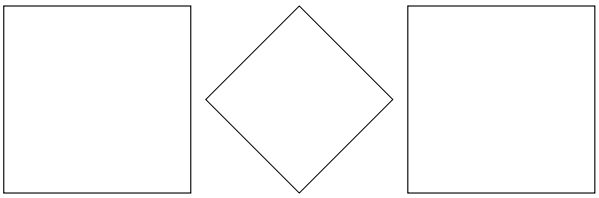
当在插图命令中指定高度或宽度时，这里给出的大小并不是图形的大小，而是图形的 BoundingBox 的大小。这点在图形旋转和缩放时很重要。例如：

```
\begin{center}
  \includegraphics[totalheight=1in]{rosette.eps}
  \includegraphics[angle=45,totalheight=1in]{rosette.eps}
  \includegraphics[angle=90,totalheight=1in]{rosette.eps}
\end{center}
```

得到



尽管看上去图形的大小不一有点奇怪，但在看过它们的 BoundingBox 后就会明白了。



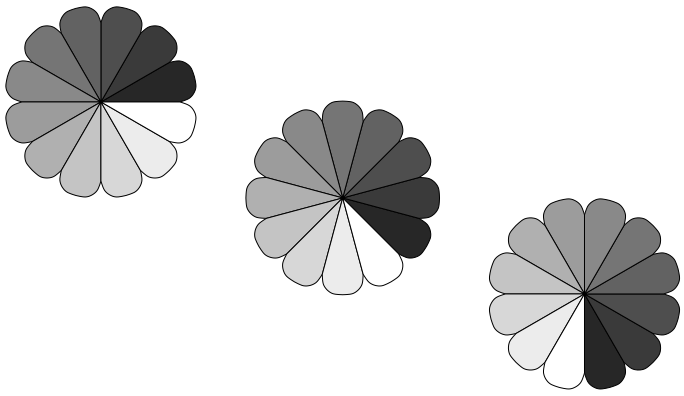
可以看到，每个图形的 BoundingBox 都被缩放到 1 英寸高。

§ 11.3. 旋转图形的对齐

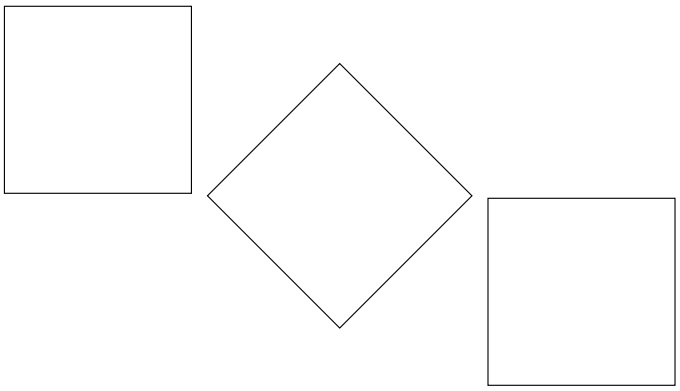
当图形被旋转时，可能会出现不对齐的情况。例如：

```
\begin{center}
   \includegraphics[totalheight=1in]{rosette.eps}
   \includegraphics[totalheight=1in,angle=-45]{rosette.eps}
   \includegraphics[totalheight=1in,angle=-90]{rosette.eps}
\end{center}
```

得到



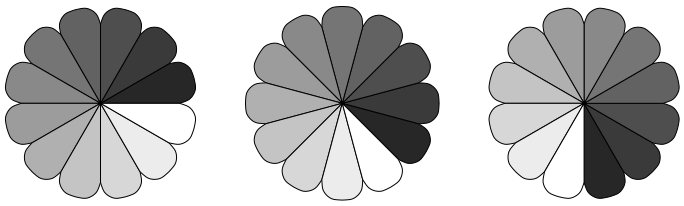
这次仍可用图形的 BoundingBox 来说明问题。



在这种情况下，我们可以看到图形对象的参考点（左下角）是处于一条水平线上的。如果希望是中间对齐，那么可以用 \includegraphics 的 origin 选项。

```
\begin{center}
   \includegraphics[totalheight=1in]{rosette.eps}
   \includegraphics[totalheight=1in,origin=c,angle=-45]{rosette.eps}
   \includegraphics[totalheight=1in,origin=c,angle=-90]{rosette.eps}
\end{center}
```

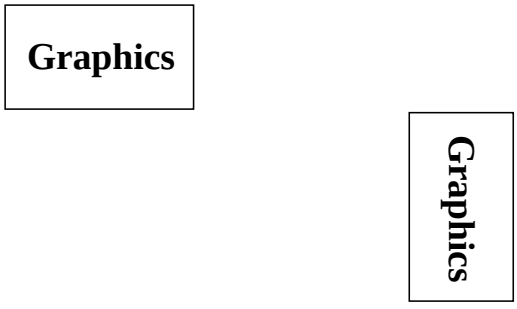
这次所有图形都是中间对齐的。



同样地，下面的命令

```
\begin{center}
   \includegraphics[width=1in]{graphic.eps}
   \hspace{1in}
   \includegraphics[width=1in,angle=-90]{graphic.eps}
\end{center}
```

将右边的图形绕它的左下角旋转，得到如下结果：



要想使图形的底部对齐，使用下面的命令：

```
\begin{center}
   \includegraphics[width=1in]{graphic.eps}
   \hspace{1in}
   \includegraphics[width=1in,origin=br,angle=-90]{graphic.eps}
\end{center}
```

上述命令让右边的图形绕它的右下角旋转，得到如下结果：



§ 11.4. 小页环境的垂直对齐

将图形放置于小页环境中是经常遇到的情况下，而且也十分有用（见第 27 章）。当小页并列时，LaTeX 会将它们的参考点垂直对齐地排列。缺省地，小页的参考点是它的左边界的中点。可用一个可选参数项来改变小页的参考点的位置。

`[b]` 使小页的参考点与小页底边的参考点对齐。

`[t]` 使小页的参考点与小页顶边的参考点对齐。

注意选项 `[b]` 不会将参考点置于小页的底部（除非其底边的参考点在它的底部），同样地，选项 `[t]` 不会将参考点置于小页的顶部（除非其顶边的参考点在它的顶部）。

当小页中只有一行时，`[b]` 和 `[t]` 选项得到的结果是一样的。如：

```
\begin{center}
  \begin{minipage}[b]{.25\textwidth}
    \centering
    \includegraphics[width=1in]{graphic.eps}
  \end{minipage}%
```

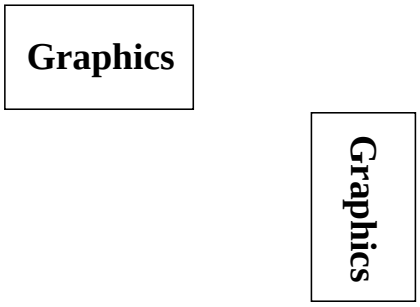


图 11.1: minipage with [b] or [t] options

```
  \begin{minipage}[b]{.25\textwidth}
    \centering
    \includegraphics[width=1in,angle=-90]{graphic.eps}
  \end{minipage}
\end{center}
```

和

```
\begin{center}
  \begin{minipage}[t]{.25\textwidth}
    \centering
    \includegraphics[width=1in]{graphic.eps}
  \end{minipage}%
  \begin{minipage}[t]{.25\textwidth}
    \centering
    \includegraphics[width=1in,angle=-90]{graphic.eps}
  \end{minipage}
\end{center}
```

都得到图 11.1 的结果。在这两种情况下，小页的参考点都是 EPS 图形的参考点（左下角）。

§ 11.4.1. 小页的底部对齐

让小页的底部对齐的一种方法是使小页的底部为其基线。如果一条高和深都为零的线在图形之后加到小页中去，那么 [b] 选项就可使小页的底部作为其基



图 11.2: Minipages with Bottoms Aligned

线。命令 `\par\vspace{0pt}` 产生那条高和深都为零的线段，这时这条深度为零的线的基线就是小页的底部，选项 `[b]` 现在让小页的底部对齐了。例如：

```
\begin{center}
  \begin{minipage}[b]{.25\textwidth}
    \centering
    \includegraphics[width=1in]{graphic.eps}
    \par\vspace{0pt}
  \end{minipage}%
  \begin{minipage}[b]{.25\textwidth}
    \centering
    \includegraphics[width=1in,angle=-90]{graphic.eps}
    \par\vspace{0pt}
  \end{minipage}
\end{center}
```

结果如图 11.2。

§ 11.4.2. 小页的顶部对齐

为使小页的顶部对齐，必须在小页的开始加入一条高度和深度都为零的线段，接着用 `[t]` 选项使得小页的基线为它的顶部。在 `\includegraphics` 前使用 `\vspace{0pt}` 加入这条高度和深度都为零的线段，由于这条线段的基线为小页顶部，所以这时 `[t]` 选项可使得小页的顶部对齐。如：

```
\begin{center}
  \begin{minipage}[t]{.25\textwidth}
    \vspace{0pt}
    \centering
```



图 11.3: Minipages with Tops Aligned

```
    \includegraphics[width=1in]{graphic.eps}
  \end{minipage}%
  \begin{minipage}[t]{.25\textwidth}
    \vspace{0pt}
    \centering
    \includegraphics[width=1in,angle=-90]{graphic.eps}
  \end{minipage}
\end{center}
```

结果如图 11.3 所示。

这里小页的顶部是和当前基线对齐。如果要求小页的顶部是和当前文本行的顶部对齐，可用 `\vspace{-\baselineskip}` 来代替 `\vspace{0pt}` 即可。这方面的问题在 [3, 第 456-457 页] 中有所涉及。

12

使 用 子 目 录

当需要大量的图形文件时，你可能希望将它们存放到一个子目录下。例如放到子目录 `images` 下，这时你试图用如下的命令来插入图形 `file.eps`。

```
\includegraphics{images/file.eps}
```

仅管这种用法在大多数 Unix 和 DOS 下的 TEX 里工作正常，它却有以下的问题：

效率不高 每当 TEX 打开一个文件，该文件名就被存入 TeX 的内存中。当打开大量的文件时，因为给出子目录名增加了文件名的长度，这种内存的占用就容易导致 poolsize 错误（见第 12.3 节）。

通用性差 LATEX 的一大优势就是它的文件能在任何操作系统平台上使用。然而，在文件名中包括子目录名会使文件依赖于操作系统，如果不作明显的改变，上面的例子就无法在 VMS 或 Macintosh 上使用。

对于图形文件存于子目录下的情形，有两种办法：

1. 最好的方法是将子目录加到 TEX 搜索路径中（见第 12.1 节）。
2. 另外一种办法是用 `\graphicspath` 命令来指明所用的子目录（见第 12.2 节）。不过，这比前一种方法的效率要低。

上述两种方法都将使 \includegraphics 自动搜索图形子目录，故可在文件中用

```
\includegraphics{file.eps}
```

来替代

```
\includegraphics{images/file.eps}
```

§ 12.1. TEX 搜索路径

因为不同的 TeX 软件设置搜索路径的方法不完全一样，所以很难提供一个普遍适用的范例。本节所用的例子是基于 Unix 下的 web2c/teTEX 的。其它版本的 TEX 也大致采用相似的策略。

对 Unix 下的 web2c/teTEX 而言，改变 TeX 的搜索路径可通过设置环境变量 TEXINPUTS 来实现。如使用 csh，

```
setenv TEXINPUTS /dir1:/dir2:
```

会使 TEX 在搜索缺省的目录前先搜索 /dir1 和 /dir2。如果省掉最后的 :，那么在搜索完 /dir1 和 /dir2 后 TEX 将不再搜索缺省的目录。如设

```
setenv TEXINPUTS :/dir1:/dir2
```

则使 TEX 在搜索缺省的目录后再搜索 /dir1 和 /dir2。而

```
setenv TEXINPUTS /dir1::/dir2
```

则使 TEX 在搜索 /dir1 后接下来搜索缺省的目录，最后再搜索/dir2。

在一个目录后面加上 // 使得此目录下的所有子目录都将被搜索。例如：

```
setenv TEXINPUTS /dir1//:/dir2:
```

会使 TEX 搜索 /dir1 的所有子目录。使用 // 要小心，如果一目录下的文件和子目录特别多的话，它会使 TEX 的搜索速度变得很慢。

若使用 sh，可用命令

```
TEXINPUTS="/dir1:/dir2:"; export TEXINPUTS
```

来设置环境变量 TEXINPUTS。

当 LaTeX 在 TeX 搜索路径中找到文件时，并不将目录名也写到 DVI 文件中，因此，旧版本的 dvips 和 xdvi 由于不会搜索 TeX 的搜索路径，可能会找不到该文件（见第 13.2 节）。

§ 12.2. 图形文件搜索路径

缺省地，LaTeX 在 TeX搜索路径中寻找图形文件。除此之外，LaTeX 还会搜索由 \graphicspath 给出的目录。例如：

```
\graphicspath{{dir1/}{dir2/}}
```

告诉 LaTeX 也从目录 dir1/ 和 dir2/ 下寻找图形文件。对 Macintosh 来说，上面的命令改为：

```
\graphicspath{{dir1:}{dir2:}}
```

很重要的一点是，搜索由 \graphicspath 给出的目录要比由 TEXINPUTS 给出的目录慢的多。更进一步说，搜索由 \graphicspath 给出的目录要占用一定的 pool space (见第 12.3节）。鉴于 \graphicspath效率不高，所以不推奖使用这一命令，最好的办法就是将要使用的目录加到 TeX 搜索路径中去（见第 12.1 节）。

§ 12.3. 节约 Pool 空间

TeX为其内部的字符串的传递保留了一部分内存空间，称为 *pool space* 。每当 TeX 打开一文件或试图打开一文件，就会有一部分 pool space 被永久性分配。当打开一个很大的文件时，这种内存的丢失会导致 TeX 耗光它的 pool size，产生如下的错误讯息：

```
! TeX capacity exceeded, sorry [poolsize=72288]
```

因为已分配的 pool space 是文件名长度的函数，所以若其中带有子目录名会使 pool space 问题更加恶化。

除了最新版的基于 web2c 的 TEX 软件和一些商业软件外，增加 poolsize 的唯一办法就是重新编译 TEX。所幸的是，通常用下面这些节约 pool space 的办法就可以解决问题。

- 避免用过长的文件名。
- 不要把子目录名包括进来

  ```
  \includegraphics{images/file.eps}
  ```

 取而代之的是将子目录加到 TEX 搜索路径中或不要把图形文件放在子目录下。
- 不要使用 \graphicspath 命令。

  ```
  \graphicspath{{dir1/}{dir2/}}
  ...
  \includegraphics{file.eps}
  ```

 将使 \includegraphics 命令试图打开下列文件：

  ```
  file.eps
  dir1/file.eps
  dir2/file.eps
  ```

 这每一次打开文件的尝试都会消耗 pool space。应该用更改 TEX 搜索路径的办法来替代使用命令 \graphicspath。
- 给出全部的文件名，不要省略文件的扩展名（特别地，像 .eps）。在缺省的 \DeclareGraphicsExtensions 定义下，命令

  ```
  \includegraphics{file}
  ```

 将使 \includegraphics 命令试图打开下列文件：

```
file.eps
file.ps
file.eps.gz
file.ps.gz
file.eps.Z
```

若是再加上使用 \graphicspath，会导致效率极低。

最好将 \DeclareGraphicsExtensions 中定义的扩展名集减到最小，这样在使用省略扩展名的文件时会好些。

13

压缩图形文件和非EPS文件的使用

当使用 dvips 时，使用者可定义一个命令来在插入图形文件之前对它进行操作。这样，如果设定此命令为一解压缩命令，就可以使用压缩的图形文件。如果设定此命令为一图形格式转换命令，就可以使用非 EPS 图形文件。考虑到目前为止 DVI 到 PS 的转换程序中只有 dvips 具有这种功能，本节所介绍的内容都需要 dvips 的支持。使用者需要在使用 graphicx 宏包时设定使用 dvips 选项。这可以通过在 \documentclass 命令中进行全局设定：

```
\documentclass[dvips,11pt]{article}
```

或者在 \usepackage 中设定 graphicx 的使用 dvips 选项为：

```
\usepackage[dvips]{graphicx}
```

推荐使用第一种方法，因为它将 dvips 这一选项传递给所有的宏包。

当使用一个支持管道[1]的操作系统时， \DeclareGraphicsRule 命令（见第 9.2 节）定义一个对文件进行操作的命令。若为解压缩命令，则可允许使用压缩的图形文件。若为图形格式转换命令，则可允许使用非 EPS 图形文件。当使用不支持管道的操作系统时，这种即时转换的命令是不允许的，这时只好将所有的图形文件都存为非压缩的 EPS 格式。

[1]例如， Unix 支持管道而 DOS 则不支持

§ 13.1. 压缩 EPS 文件的例子

使用压缩 EPS 文件的步骤是：

1. 创见一个 EPS 文件（比如说 `file.eps`）。
2. 将它的 BoundingBox 存放到另外一文件中（ `file.eps.bb`）。
3. 压缩 EPS 文件，比如用 Unix 命令：

```
gzip -9 file.eps
```

得到压缩文件 `file.eps.gz`。这里 `-9`（或者 `-best`）选项表示最佳压缩。

4. 在 `\includegraphics`前声明适当的 `\DeclareGraphicsRule` 命令。使得 LaTeX 知道如何处理特殊后缀的文件（见第 9.2 节）。例如：

```
\documentclass[dvips]{article}
\usepackage{graphicx}
\begin{document}
\DeclareGraphicsRule{.eps.gz}{eps}{.eps.bb}{‘gunzip -c #1}
\begin{figure}
  \centering
  \includegraphics[width=3in]{file1.eps.gz}
  \caption{Compressed EPS Graphic}
  \label{fig:compressed:eps}
  \end{figure}
\end{document}
```

在这个特殊的例子里， `\DeclareGraphicsRule` 实际上是可以省略的，因为在 `dvips.def` 已经定义过了。如果使用另外一个解压缩程序或文件名后缀，那么 `\DeclareGraphicsRule` 是不能少的。例如 BoundingBox 存放到文件 `file.bb` 中，则相应的 `\DeclareGraphicsRule` 应为：

```
\DeclareGraphicsRule{.eps.gz}{eps}{.bb}{‘gunzip -c #1}
```

§ 13.2. TEX 搜索路径和 dvips

当 LATEX 遇上一 \includegraphics 命令时，会首先在当前目录下搜寻图形文件。如果找不到所需的文件，LATEX 将按照 TEX 搜索路径来寻找。当 DVI 文件转为 PS 文件时，dvips 也是同样地顺序来搜寻图形文件。这不会有什么问题。然而，如果用 \DeclareGraphicsRule 定义了一个即时转换的命令，那么此命令将会阻止 dvips 在 TEX 搜索路径中寻找图形文件。例如：

```
\DeclareGraphicsRule{.eps.gz}{eps}{.eps.bb}{‘gunzip -c #1}
```

指定对后缀为 .eps.gz 的文件使用命令 gunzip -c。假设用下面的命令来插入图形文件，

```
\includegraphics{file.eps.gz}
```

那么若 file.eps.gz 和 file.eps.bb 在当前目录下的话，一切都会很顺利。LATEX 使用 file.eps.bb 而 dvips 使用 gunzip -c file.eps.gz 来解压缩图形文件。

但是，如果 file.eps.gz 和 file.eps.bb 不在当前目录下，而是在目录 /a/b/c/ 下（假设该目录已加到 TEX 搜索路径中）。LATEX 仍然能够找到 /a/b/c/file.eps.bb，但 dvips 在执行 gunzip -c file.eps.gz 就会出问题。因为 gunzip 找不到 file.eps.gz。假如你的 TEX 软件使用了 kpathsea 库（比如 teTeX），这个问题可用定义下面的图形规则来解决。

```
\DeclareGraphicsRule{.eps.gz}{eps}{.eps.bb}%
                    {‘gunzip -c ‘kpsewhich -n latex tex #1‘}
```

这里使用 \kpsewhich 来为 gunzip 找寻文件。‘kpsewhich -n latex tex #1 使得 dvips 在 TEX 搜索路径中寻找压缩图形文件，然后把文件的全名（包括目录名）附加到 gunzip -c 命令后，使得即使压缩图形文件不在当前目录下，gunzip 也可对其进行操作。

虽然上面给出的新的图形规则可以放在 LATEX 文件的开头，但是最好的用法是把它放到 graphics.cfg 文件中：

```
\AtEndOfPackage{%
\DeclareGraphicsRule{.eps.gz}{eps}{.eps.bb}%
                    {`gunzip -c `kpsewhich -n latex tex #1`}}
```

并且保留 \ExecuteOptionsdvips 这一行。

旧版本的 dvips

因为旧版本的 dvips 不会搜索 TEX 搜索路径， dvips 无法找到位于 TEX 搜索路径中的文件，下面的命令利用 kpsewhich 为 dvips 搜索位于 TEX 搜索路径中的非压缩的 EPS 文件。

```
\DeclareGraphicsRule{.eps}{eps}{.eps}%
                    {`cat `kpsewhich -n latex tex #1`}
```

（当然最好的解决办法是升级你的 TEX 软件。）

§ 13.3. 非 EPS 图形文件

EPS 格式的图形文件可以很容易的插入到 LATEX 文件中，而非 EPS 格式的图形文件则不是将插图命令中的文件名替换一下就可以的。对于不同的图形驱动来说，所支持的图形格式也不尽相同。而不同版本的 TEX 软件也有各自支持的非 EPS 格式图形。一般来说，除了 .png 格式的图形文件外，其它的非 EPS 格式图形基本上只有一两种图形驱动支持直接使用它们。更多的情况是需要先转换为 EPS 格式的图形文件[2]再插入到 LATEX 文件中。这样就要求有相应的图形格式转换工具。尽管使用非 EPS 格式的图形文件不如 EPS 图形文件简单方便，但由于它们可能比 EPS 文件要小，而一些绘图软件也不能生成 EPS 文件，所以有时还是希望在 DVI 文件转换为 PS 文件时再对其进行格式转换。如果使用 dvips，这种即时转换的命令可用 \DeclareGraphicsRule来给出。例如用这种方法将 file2.gif 加到 LATEX 文档中需要以下几步：

1. 找到一个支持命令行方式的 GIF 到 EPS 的转换工具（假设为 gif2eps）。
2. 建立一个注明 file2.gif 自然大小的 BoundingBox 文件。为此，
 (a) 用 ebb file2.gif 直接得到 BoundingBox 文件[3]。

[2] 在使用 pdfLATEX 和 dvipdfm 图形驱动时不支持 EPS 格式的图形，而是支持 PDF 格式的图形。
[3] ebb 是 dvipdfm 中的一个用来计算非 EPS 图形文件的 BoundingBox 应用程序。

(b) 将 file2.gif 转为 PostScript 文件，若其中有 BoundingBox 行，则将此行存放到文件 file2.gif.bb 中，否则，可按照第 3.3 节的方法来计算 BoundingBox 并将所得到的结果放在 file2.gif.bb 中的 %%BoundingBox: 后。然后将 PostScript 文件删除。

3. LaTeX 文件中，在 \includegraphics 命令前，加入图形规则：

```
\DeclareGraphicsRule{.gif}{eps}{.gif.bb}{`gif2eps #1}
```

当遇到 \includegraphics{file.gif} 时，LaTeX 从 file.gif.bb 中读取 BoundingBox 并告诉 dvips 使用 gif2eps 来将 file2.gif 转为 EPS 文件。

§ 13.3.1. GIF 的例子

由于插入非 EPS 格式的图形所需的命令依赖于操作系统和图形格式转换程序，在此提供两个 Unix 系统下常用的转换程序的例子。

```
\DeclareGraphicsRule{.gif}{eps}{.gif.bb}{`convert #1 'eps:-' }
\begin{figure}
  \centering
  \includegraphics[width=3in]{file2.gif}
  \caption{GIF Graphic}
\end{figure}
```

这里使用 convert（包含在 ImageMagick 中）来将 GIF 转为 EPS。而命令：

```
convert file2.gif 'eps:-'
```

将 file2.gif 转为 EPS 格式的图形并输出到标准输出。

另一方法是使用 giftoppm，ppmtopgm 和 pgmtops 来将 GIF 转为 EPS。只需在上例中将图形规则改为：

```
\DeclareGraphicsRule{.gif}{eps}{.gif.bb}%
                    {`giftoppm #1 | ppmtopgm | pgmtops}
```

§ 13.3.2. 对非 EPS 图形的直接支持

虽然 LaTeX 和 dvips 不断地被要求直接支持非 EPS 图形并使得如同 EPS 图形一样简单方便。的确，这样做会带来不少方便，但却存在着不少问题。

- 因为 LaTeX 是通过从 EPS 文件中读取 BoundingBox 来确定图形文件的大小的，加上 LaTeX 只能读取 ASCII 文件，所以其它的非 EPS 图形文件（绝大多数是二进制文件）会阻碍 LaTeX 获取图形大小的信息。
- 进一步讲，支持非 EPS 图形要求 dvips 具有图形格式转换的能力（GIF-to-PS, TIFF-to-PS, 等）。这需要大量的编程和维护工作。

有鉴于此， dvips 提供调用外部图形转换程序的机制而不是直接支持非 EPS 图形文件。这种机制允许 LaTeX 通过设置 \DeclareGraphicsRule 来使 dvips 调用指定的外部图形转换程序。这样使用者可自己选择图形转换程序， dvips 也不用捆绑一些图形转换功能，从而比直接支持非 EPS 图形文件更具灵活性。

仅管 LaTeX 和 dvips 一般不支持直接插入非 EPS 的图形，也还是有几个例外：

1. 如果 dvips 编译时用了参数 -Demtex，它将支持一些 EmTeX 的 \special 命令，允许直接插入 PCX, BMP 或 MSP 位图。
2. Macintosh 下的共享 TeX/LaTeX 软件 Oztex2.1 中， DVI 到 PS 的转换程序 OzDVIPS 允许通过 \special 命令来使用 MacPaint 和 PICT 文件。详见

 http://www.kagi.com/authors/akt/oztex.html
3. 一些商业版本的 LaTeX 支持非 EPS 的图形。
 (a) Macintosh 下的 Textures 支持 PICT 图形。详见

 http://www.bluesky.com/

 (b) Y&Y 的 Windows 版本的 TeX 中， DVI 到 PS 的转换程序 DVIPSONE 支持 TIFF 图形。详见

 http://www.YandY.com/

即使上述方法中，TEX 仍然无法直接从二进制的图形文件中获得其图形的大小。为使 LATEX 能正确地给所插入的图形分配空间，使用者必须用 .bb 文件或在 \includegraphics 中用 bb 选项给出图形的大小。

14

Psfrag 宏 包

目前大多数绘图和分析软件都可以输出 EPS 格式的图形，但是它们大都不能像 LaTeX 一样支持符号和公式。 PSfrag 宏包允许用 LaTeX 的文本和公式来替代 EPS 图形文件中的字符。在 CJK 等中文环境下，可以使用 PSfrag 将图形中的标记字符替换为所需的中文文本。

PSfrag 3.0 是 1996 发布的正式版本，几乎是被完全重新写过。以前的版本则需要借助预处理程序（`ps2frag` 或 `ps2psfrag`）来识别和记录 EPS 图形文件中的文本。而 PSfrag 3.0 不需要借助预处理程序，也不需要像 `perl` 或 `ghostscript` 等外部程序。 PSfrag 3.0 只需要较近版本的 LaTeX（*12/95* 或以后）和 LaTeX 图形宏包套件。参考文献 [7] 给出了 PSfrag 3.0 的详细说明。

新版的 PSfrag 3.0 的另一优势是支持压缩的 EPS 图形。不过， `\tex` 命令（见第 14.3 节）不能被用来在压缩的 EPS 图形中嵌入 LaTeX 文本。

为使用 PSfrag，生成一 EPS 图形文件，然后按照以下步骤：

1. 在 LaTeX 文档的导言区中加入：`\usepackage{psfrag}`。
2. 在 LaTeX 文档中，使用 `\psfrag` 命令来指明那些 EPS 图形中的文本将被什么样的 LaTeX 文本所替代。这些替换会在同一环境下后面的任何 `\includegraphics` 命令中执行。
3. 像通常一样使用 `\includegraphics`

表 14.1: PSfrag Options

PStext	EPS 图形中被替换的文本。
posn	（可选项，缺省为 *[Bl]*）放置点相对于 LaTeX 文本的参考位置。
PSposn	（可选项，缺省为 *[Bl]*）放置点相对于现存的 EPS 文本的参考位置。
scale	（可选项，缺省为 1）LaTeX 文本的缩放因子。为得到最好的效果，建议不使用这一选项，而使用 LaTeX 的字体命令如 \small 和 \large 等。
rot	（可选项，缺省为零）当给出一个角度时，此角度即为新的 LaTeX 文本相对于旧的 EPS 图形中文本的角度。它以度为单位并且逆时针方向为正。此选项对处理那些由只允许水平方向的文本的应用软体生成的 EPS 图形时特别有用。
text	用来替换旧的 EPS 图形中文本的 LaTeX 文本。如同通常的 LaTeX 文本，数学公式必须放在美元符号对中。如：`$\frac{1}{2}$` 或 `$x^2$`。

\psfrag 命令的用法如下：

```
\psfrag{PStext}[posn][PSposn][scale][rot]{text}
```

上面命令中的参数的说明见表 14.1。 posn 和 PSposn 选项可为第 32 页图 8.1 所示的 12 个点中的一个。如果没有给出，则缺省为 [Bl]。空的选项则设定为 c（如 [] 就等于 [c]， [l] 就等于 [lc]）。可参考 [7] 中各种位置组合的例子。

注意 \psfrag 只匹配整个字符串，如下面的命令

```
\psfrag{pi}{$\pi$}
```

用 π 替换 pi。但却不会替换 EPS 文件中的其它像 pi/2 或 2pi 这样的字符串。对于这样的字符串必须分别使用 \pafrag 命令。

如果所替换 EPS 中字符串不是完整的置于一 PS 命令中， PSfrag 将不起作用。在一些应用软件生成的 EPS 图形中，为达到特殊的字符间距，将一字符串分隔为几个子串或单个的字符。例如， Corel Draw 用如下的 EPS 代码来放置字符串 “Hello World”：

```
0 0 (Hello W) @t
1080 0 (orld) @t
```

由于 PSfrag 把它看作是两个不相干的字符串 “Hello W” 和 “orld”，所以任何对 “Hello World” 的替换都不起作用。如果不能在应用软件中取消这种对字符间距的处理，使用 Courier 或其它单一间距的字体一般可防止这种情况。如果确实无法避免这种情况，那么只能对单个字符进行替换。

§ 14.1. Psfrag 使用例一

命令

```
\includegraphics{pend.eps}
```

只是插入 EPS 图形而没有任何 PSfrag 替换，见图 14.1。而下面的命令

```
\psfrag{q1}{$\theta_1$}
\psfrag{q2}{$\theta_2$}
\psfrag{L1}{$L_1$}
\psfrag{L2}{$L_2$}
\psfrag{P1}[][]{$P_1$}
\psfrag{P2}[][]{\large $P_2$}
\includegraphics{pend.eps}
```

插入 EPS 图形，并使用 PSfrag 对 EPS 图形中的字符串进行替换，见图 14.2。前面四个 \psfrag 命令中，新的 LaTeX 字符串的左基线点对应于旧的 EPS 字符串的左基线点，后面两个 \psfrag 命令中使用 [][] 选项使得新的 LaTeX 字符串的中心对应于旧的 EPS 字符串的中心。注意并不是所有的 EPS 字符串都被替换了，如在图 14.2 中 N 就没有被替换。

§ 14.2. Psfrag 使用例二

这个例子演示了 \shortstack, \colorbox 和 \fcolorbox 等命令如何与 \psfrag 一起使用。

\shortstack 这一命令允许将文本竖直放置，每行用 \\ 分开。可用来使用多行文本替换 EPS 图中的一行文本。

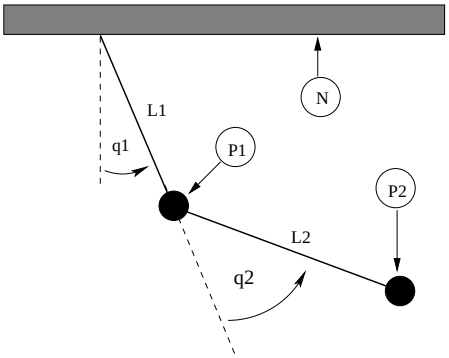


图 14.1: Without PSfrag Replacement

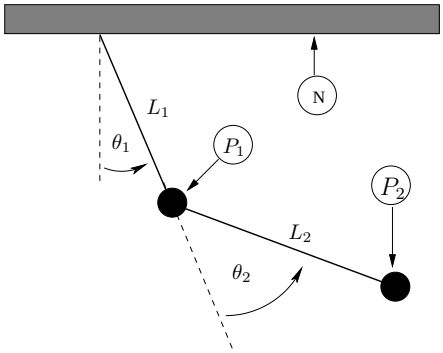


图 14.2: With PSfrag Replacement

`\colorbox` color 宏包所提供的一个命令，它在所作用的对象背后放置一长方形的彩色区域作为背景。此背景超出对象的部分的大小由长度 `\fboxsep` 控制。例如：

```
\colorbox{yellow}{文本}
```

在 文本 后放置了一长方形的黄色背景。有关 `\colorbox` 的详细说明可参考文献 [5]。

在使用 PSfrag 时， `\colorbox` 常常用来放置那些由于线条或阴影而被遮挡的文本。通过将这些文本的背景色设为白色，防止它们被图形所遮挡。

`\fcolorbox` 这一命令（也由 color 宏包所提供）与 `\colorbox` 类似，只是为背景加上了一个边框。如命令

```
\fcolorbox{black}{yellow}{文本}
```

在 文本 后放置了一长方形的带有黑色边框的黄色背景。

这里边框的宽度由 `\fboxrule` 控制，边框和对象之间的间隔大小则由 `\fboxsep` 控制。

图 14.3 和图 14.4 显示了这些命令与 PSfrag 配合使用的效果。图 14.3 是没有使用 PSfrag 的原始图形，而图 14.4 则是使用如下命令的结果。

```
\psfrag{q1}[][]{\colorbox{white}{$q_1$}}
\psfrag{base}{\fcolorbox{black}{white}{Base}}
```

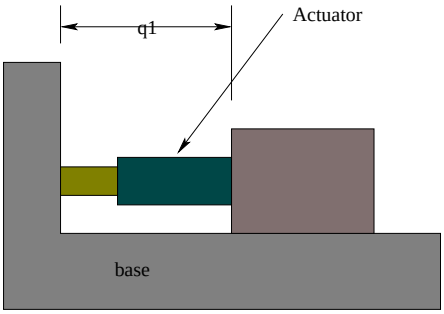


图 14.3: Without PSfrag Replacement

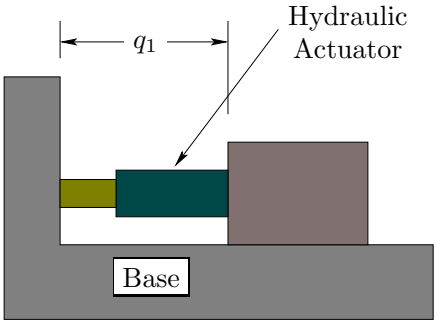


图 14.4: With PSfrag Replacement

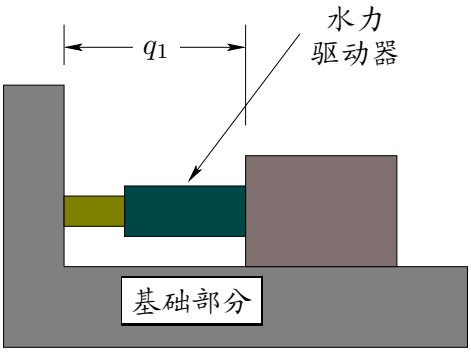


图 14.5: PSfrag 中使用中文的例子

```
\psfrag{Actuator}[l][l]{\shortstack{Hydraulic\\ Actuator}}
\includegraphics{mass.eps}
```

下面的例子使用了中文，结果如图 14.5。

```
\psfrag{q1}[][]{\colorbox{white}{$q_1$}}
\psfrag{base}{\fcolorbox{black}{white}{基础部分}}
\psfrag{Actuator}[l][l]{\shortstack{水力\\ 驱动器}}
\includegraphics{mass.eps}
```

§ 14.3. EPS 图形中的 LaTeX 文本

在使用 PSfrag 宏包时， \psfrag 命令是最常使用也是推荐使用的。它的具体用法已在前面几节中介绍过了。此外， PSfrag 宏包还提供 \tex 来直接将

LaTeX 文本嵌入到 EPS 中。不过，它的效率要比 \psfrag 低。有关 \tex 的详细信息可参见 [7]。

§ 14.4. 图形和文本的缩放

如果一幅使用了 PSfrag 的图形被缩放，那么 PSfrag 所替换的文本也相应的被缩放。因此，使用 graphicx 包时的一些细节有可能影响到这些文本的大小。

- 当使用 `width`，`height` 或 `totalheight` 来指定图形的大小，如

  ```
  \includegraphics[width=3in]{file.eps}
  ```

 这时 PSfrag 所替换的文本在 `file.eps` 被缩放到 3 英寸后才加入。相反，

  ```
  \resizebox{3in}{!}{\includegraphics{file.eps}}
  ```

 则是将图形 `file.eps` 以它的自然大小插入，进行 PSfrag 替换，然后再将图形和所替换的文本一起缩放到 3 英寸。

- 相似地，当缩放选项在旋转选项之前时，

  ```
  \includegraphics[width=3in,angle=30]{file.eps}
  ```

 缩放选项会得到预期的效果。然而，当它在旋转选项之后时，

  ```
  \includegraphics[angle=30,width=3in]{file.eps}
  ```

 图形 `file.eps` 会先以其自然大小插入，然后被旋转，接着被缩放到 3 英寸，因为 PSfrag 的替换是在图形被插入时发生的，所以第二个命令中的替换文本会被缩放，而第一个命令中的替换文本不会被缩放。如果图形的自然大小和被缩放后的大小差别很大的话，这两个命令得到的结果会大不相同。

参见 [7] 以获取关于 PSfrag 的详细说明。

§ 14.5. Psfrag 的不兼容性

虽然 PSfrag 3.0 比其以前的版本有很多优越性，但它却与 Xfig 生成的使用了填充模式对象的 EPS 图形不兼容。 PSfrag 包中的 readme.xfg 描述了这种不兼容性。关于此问题的解决办法将在第 14.5.1 节中加以讨论。

PSfrag 包中的另一文件 readme.sem 描述了 PSfrag 和 Seminar 宏包之间的不兼容性。值得庆幸的是，在 CTAN 的最新版本的 Seminar 宏包已消除了这一不兼容性。

§ 14.5.1. Xfig EPS 文件

Xfig 生成的使用了 “pattern fill” 的 EPS 图形文件在与 PSfrag 配合使用时会遇到问题。问题的原因在于 Xfig 和 PSfrag 都对 PostScript 命令 show 进行重新定义。按说这种重新定义不应该互相冲突，事实上却非如此。

尽管 PSfrag 的维护者们还没有确定一个长远的解决办法，但是下面所提供的方法似乎可以解决这一问题。

1. 在 EPS 文件中，找到 /PATfill 命令。
2. 在 /PATfill 命令的定义中，找到 show 命令。这里 show 命令只会出现一次。
3. 将 show 用 oldshow 来替代（ oldshow 置于 XFig 存放 “old” 版本的显示机制的地方，在它因自己的目的来重定义 show 之前。）。

如果你发现 PSfrag 或 Xfig 对此有另外的解决办法，请和 PSfrag 的维护者们联系（ psfrag@rascals.stanford.edu ）。

§ 14.6. Overpic 宏包

尽管 PSfrag 的功能十分强大，使用起来也很方便，但对于非 EPS 图形或标记并非标准的字符串的 EPS 图来说，它就不能被使用。此外，由于像一些新的 TEX 软件如 pdfLATEX 等不能直接使用 EPS 图形，也限制了 PSfrag 的使用。本

节所介绍的 overpic 宏包允许直接将 LaTeX 对象放置到一幅图形上，而不是通过对图形上已有的标记进行替换来实现。这样，虽然在定位时要麻烦一些，却可以在一些不能使用 PSfrag 的情况下得到同样的效果。

overpic 宏包中定义了一个 `overpic` 环境，它有效地将 `picture` 环境和 `\includegraphics` 命令结合起来。使得 `picture` 环境的维数和插入的 EPS 图形的维数相同。这样就可以很容易地把 LaTeX 的命令放到图形上的任何指定位置。同时，还可以在图形上加上标尺以方便定位。

```
\begin{overpic}[选项]{图形}<LaTeX 对象>\end{overpic}
```

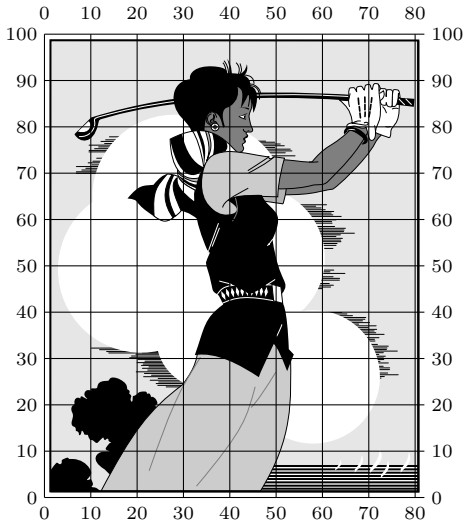
这里的选项可为 `scale`，`grid`，`tics`，`unit` 等。分别表示对图形进行缩放，加标尺，设定度量单位等。在调入 textsfoverpic 宏包时，若使用参数 `abs`，即

```
\usepackage[abs]{overpic}
```

则在 `overpic` 环境中使用绝对位置。即放置 LaTeX 对象的位置以实际度量来定位。若使用

```
\usepackage{overpic}
```

则在 `overpic` 环境中使用相对位置。即放置 LaTeX 对象的位置以其相对于图形大小的百分比来定位。下面是几个例子（使用相对位置）：



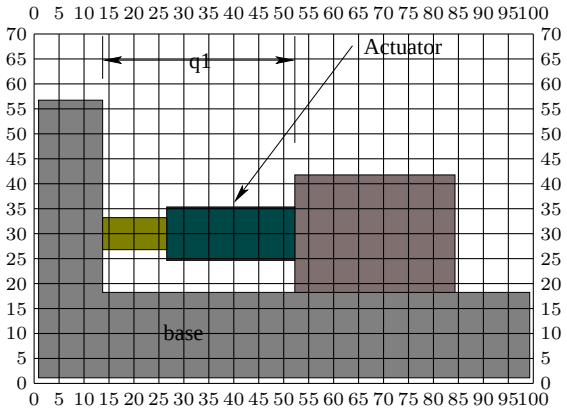

```
\begin{overpic}[scale=.25,grid,tics=10]%
                {golfer.ps}
\end{overpic}
```




```
\begin{overpic}[scale=.25]{golfer.ps}
  \put(5,50){\LaTeX}
  \put(5,40){\color{red}外部图形}
  \put(55,10){%
    \includegraphics[scale=.07]%
                    {golfer.ps}}
```

对于第 14.2 节中的例子，现在使用 overpic 来得到同样的结果。首先可使用 grid 和 tics 选项来确定放置 LaTeX 对象的位置（这样做只是为了能够得到更精确的放置位置，在定位后就可将 grid 和 tics 选项去掉）。

```
\begin{overpic}[scale=1.2,grid,tics=5]{mass.eps}
  \end{overpic}
```



根据上图来将所需的 LaTeX 对象放到图形上的合适位置。

```
\begin{overpic}[scale=1.2]{mass.eps}
  \put(25,8){\fcolorbox{black}{white}{基础部分}}
  \put(31,64){\colorbox{white}{$q_1$}}
  \put(65,65){\colorbox{white}{\shortstack{水力\\ 驱动}}}
\end{overpic}
```

结果如图 14.6所示。

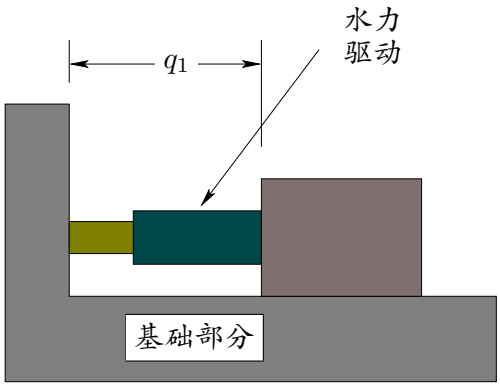


图 14.6: Overpic example

15

多次使用同一图形的几种技巧

当一幅图形在文档中被多次使用时，它的 EPS 代码将会多次出现在最后得到的 PS 文件中。特别地，譬如在文档的页眉或页脚使用标志或其它图形时，就会遇到这种情形。本章将介绍一些多次使用同一图形的技巧。

一般来说，多次使用同一图形通常有以下四种方法：

1. 每次使用图形时均用 `\includegraphics{file.eps}`。但这样做会有两个问题：
 (a) 每次用到 `\includegraphics` 时， LaTeX都得搜索和打开一次图形文件。
 (b) 在最后生成的 PS 文件中， EPS 图形代码多次出现，使生成的 PS 文件变得很大。
2. 将 EPS 图形文件存放到一个 LaTeX 盒子中，每当用到图形时就调用这一个 LaTeX 盒子来插入图形。这将使 LaTeX 只需搜索和打开一次图形文件即可。在 LaTeX 文件的开头加入命令：

```
\newsavebox{\mygraphic}
\sbox{\mygraphic}{%
      \includegraphics{file.eps}}
```

每次使用图形时，用命令 \usebox{\mygraphic}。图形的缩放和旋转可用 \scalebox 和 \rotatebox 来得到。不过，在最后生成的 PS 文件中， EPS 图形代码仍会多次出现。 PS 文件的大小没有改变。

3. 当 EPS 的图形是一个矢量图形时，可将此绘图的代码定义为一个 PostScript 命令，当用到图形时就调用这一个 PostScript 命令。详见第 15.1 节。因为在最后生成的 PS 文件中只包含了一次 EPS 的图形代码，所以 PS 文件会很小。不过在打印时由于绘图命令一直存放在打印机的内存中，很容易导致打印机的内存耗尽而无法打印。另外，使用这种方法时， LaTeX 仍然得每次对图形文件进行搜索和打开操作。

4. 像前面一种方法一样定义 PostScript 绘图命令，但把它存放到一个 LaTeX 盒子中。这样不仅最后生成的 PS 文件很小，而且 LaTeX 也只需搜索和打开一次图形文件即可。

§ 15.1. 定义 PostScript 命令

本节介绍如何定义一个 PostScript 命令来完成一幅 EPS 矢量图形的绘图指令。这一方法不适合于那些包含位图的 EPS 图形。

为了将一 EPS 图形转化为一 PostScript 命令，必须将 EPS 图形文件分为两个文件。其中一个定义了 PostScript 字典和图形命令，另一个则含有图形文件信息和使用已定义的 PostScript 命令。例如，一个用 xfig 生成的 EPS 文件有如下的形式：

```
%!PS-Adobe-2.0 EPSF-2.0
%%Title: /tmp/xfig-fig017255
%%Creator: fig2dev Version 2.1.8 Patchlevel 0
%%CreationDate: Sun Sep 3 15:36:01 1995
%%Orientation: Portrait
%%BoundingBox: 0 0 369 255
%%Pages: 0
%%EndComments
/$F2psDict 200 dict def
$F2psDict begin
```

```
...
%%EndProlog
$F2psBegin
...
$F2psEnd
```

这里 ... 代表没有列出的命令。一个 EPS 文件一般包括三部分：

1. 以 % 开始的 header 命令。
2. Prolog 部分开始于

   ```
   /$F2psDict 200 dict def
   ```

 结束于

   ```
   %%EndProlog
   ```

 Prolog 部分定义了 EPS 文件所使用的 PostScript 字典中的命令。在这个例子中， PostScript 字典名为 $F2psDict，当然不同的文件可有不同的名字。
3. 最后一部分包含用来绘图的命令。

假设上面的这个 EPS 文件名为 file.eps。新建两个文件，分别命名为 file.h 和 file.ps。其中 file.h 含有如下内容：

```
/$F2psDict 200 dict def
$F2psDict begin
...
%%EndProlog
/MyFigure {
$F2psBegin
...
$F2psEnd }
def
```

file.ps 含有如下内容：

```
%!PS-Adobe-2.0 EPSF-2.0
%%Title: /tmp/xfig-fig017255
%%Creator: fig2dev Version 2.1.8 Patchlevel 0
%%CreationDate: Sun Sep 3 15:36:01 1995
%%Orientation: Portrait
%%BoundingBox: 0 0 369 255
%%Pages: 0
%%EndComments
$F2psDict begin MyFigure end
```

`file.h` 中定义了 PostScript 字典和命令 `/MyFigure`， `file.ps` 则包含了 EPS 文件的 `header` 信息，并且使用了 `file.h` 中定义的 PostScript 命令。特别指出的是， `file.ps` 中包含有 `%!PS...` 行和 BoundingBox 行是非常重要的。这时，可像下面的例子一样在 LaTeX 文件中使用这个图形了。

```
\documentclass{article}
\usepackage{graphicx}
\special{header=file.h}
\begin{document}
  ...
  \includegraphics[width=2in]{file.ps}
  ...
  \includegraphics[totalheight=1in]{file.ps}
  ...
\end{document}
```

注意原始图形文件 `file.eps` 并没有被使用。因为 `file.h` 中的 PostScript 命令只被使用了一次，所以最后得到的 PS 文件很小。然而，每次插入图形的时候，LaTeX 都得搜索和读取 `file.ps` 一次。下面的命令将图形存放到一个 LaTeX 盒子中，使得在 LaTeX 只搜索和读取 `file.ps` 一次的情况下仍得到很小的 PS 文件。

```
\documentclass{article}
\usepackage{graphicx}
\special{header=file.h}

\newsavebox{\mygraphic}
  \sbox{\mygraphic}{%
```

```
    \includegraphics[width=2in]{file.ps}}

\begin{document}
  ...
  \usebox{\mygraphic}
  ...
  \resizebox*{1in}{!}{\usebox{\mygraphic}}
  ...
\end{document}
```

所得的结果和上一个领子是一样的。

§ 15.2. 在页眉和页脚使用图形

在页眉和页脚使用图形的一个最容易的方法是使用 fancyhdr（它是旧的 fancyheadings 的增强版本）。fancyhdr 的用法和宏包说明详见文献 [12]。

在 LaTeX 文档中，页眉由左、中、右三部分组成。`\fancyhead` 命令指定了页眉的形式和内容，并以 `L,C,R` 区分左、中、右区域。例如：

```
\pagestyle{fancy}
\fancyhead[C]{我的文档}
```

使得页眉的中间部分印出“我的文档”，而

```
\pagestyle{fancy}
\fancyhead[L,R]{\textbf{Confidential}}
```

使得页眉的左右都印出“**Confidential**”。如果没有指定 `L,C,R` 中的任何一个，那么由 `\fancyhead` 定义的内容将在三个区域中都会印出。相似地，`\fancyfoot` 则用来定义页脚的左、中、右三个区域。

页眉和页脚的图形

可以利用 fancyhdr 宏包中的命令来 在页眉和页脚上使用图形。例如，在用第 15.1 节的方法将 EPS 文件 `file.eps` 分为 `file.h` 和 `file.ps` 后，下面的命令

```
\documentclass{article}
\usepackage{fancyhdr,graphicx}
```

```
\renewcommand{\headheight}{0.6in}% must be large enough for graphic
\renewcommand{\textheight}{7.5in}

% Define PostScript graphics command
\special{header=file.h}

% Save graphics in LaTeX box
\newsavebox{\mygraphic}
\sbox{\mygraphic}{\includegraphics[totalheight=0.5in]{file.ps}}
\pagestyle{fancy}
\fancyhead{} % clear all header fields
\fancyhead[L]{\usebox{\mygraphic}}
\fancyfoot{} % clear all footer fields
\fancyfoot[C]{\thepage}
\renewcommand{\headrulewidth}{0.5pt}
\renewcommand{\footrulewidth}{0pt}
\begin{document}
  ...
\end{document}
```

将图形放置在每一使用 “fancy” 风格页的左上角，并且下面有一条宽为 0.5pt 的横线。此外，每页的页角的中央放置页码，但它的上方没有横线。这些设置不会影响 “plain” 风格的页面。

奇/偶页的页眉

当使用 `[twoside]` 排版选项时，经常希望在奇数页和偶数页设置不同页眉和页脚，这时可使用 `O,E` 选项来区分奇数页和偶数页。如果没有给出 `O,E` 选项，则页眉和页脚的命令会应用到所有的页面中，无论是奇数页还是偶数页。例如：

```
\pagestyle{fancy}
\fancyhead[LE]{我的文章}
\fancyhead[RO]{我的名字}
\fancyfoot[C]{\thepage}
```

在偶数页的左上角放置 我的文章，在奇数页的右上角放置 我的名字，页脚的中央则放置页码。而命令

```
\pagestyle{fancy}
\fancyhead[LE,RO]{\usebox{\mygraphic}}
\fancyfoot[C]{\thepage}
```

使得偶数页的左上角和奇数页的右上角印出图形。

改变 Plain 页面式样

\fancyhead 命令只对那些页面式样为 “fancy” 的页面起作用。即使用 \pagestyle{fancy} 将文档的页面式样设置为 “fancy” 式样，一些页面，如封面，目录和每章的第一页仍为缺省的 “plain” 式样。

改变 Plain” 页面式样的缺省设置可用 \fancypagestyle 命令来实现。例如将下面的命令加到上面的例子中可使得封面，目录等的页眉上也将会有图形印出。

```
\fancypagestyle{plain}{%
  \fancyhead{} % clear all header fields
  \fancyhead[L]{\usebox{\mygraphic}}
  \fancyfoot{} % clear all footer fields
  \fancyfoot[C]{\thepage}
  \renewcommand{\headrulewidth}{0.5pt}
  \renewcommand{\footrulewidth}{0pt}}
```

当使用 [twoside] 排版选项时，将上面的

```
\fancyhead[L]{\usebox{\mygraphic}}
```

替换为

```
\fancyhead[LE,RO]{\usebox{\mygraphic}}
```

则在每一页的页眉上都放置上图形。

§ 15.3. 在背景中使用图形水印

有时在排版时会遇到在背景中使用图形水印的情况。如同上节讨论的那样，也可用 fancyhdr 来实现。下面的例子中利用 fancyhdr 宏包中的命令，在每一页都用图形 file.eps 作为背景。

```
\documentclass{article}
\usepackage{graphicx,fancyhdr}
%%% store graphics in a box
```

```
\newsavebox{\mygraphic}
  \sbox{\mygraphic}{%
    \includegraphics[keepaspectratio, height=0.8\textheight,%
                     width=0.8\textwidth]{file.eps}}
\pagestyle{fancy}
\fancyhead{}
\fancyhead[C]{\setlength{\unitlength}{1in}
              \begin{picture}(0,0)
              \put(-2.2,-6){\usebox{\mygraphic}}
              \end{picture}}
\fancypagestyle{plain}{%
  \fancyhead{}%
  \fancyhead[C]{\setlength{\unitlength}{1in}
                \begin{picture}(0,0)
                \put(-2.2,-6){\usebox{\mygraphic}}
                \end{picture}}}
\begin{document}
  ...
\end{document}
```

在上面的例子中，图形的放置位置在页眉中央下方 6 英寸，偏左 2.2 英寸的地方。可通过改变上述参数来改变图形的放置位置。

因为页眉在正文文本之前排出，所以正文文本会出现在图形的上方，从而得到水印的效果。反之，因为页脚在正文文本之后排出，所以若在页脚中使用图形会覆盖正文文本。如果 `file.eps` 是一矢量图形，可用第 15.1 节的方法来使得最后生成的 PS 文件较为小些。

另一种得到图形水印效果的方法是使用 eco-pic 宏包。该宏包可从 CTAN 下载。它提供命令 `\AddToShipoutPicture` 把任何 LaTeX 图形环境先于正文文本排出，从而得到水印的效果。使用 eso-pic 的例子：

```
\begin{document}
\usepackage{graphicx}
\usepackage{eso-pic}

%%% store graphics in a box
\newsavebox{\mygraphic}
```

```
  \sbox{\mygraphic}{%
    \includegraphics[keepaspectratio, height=\textheight,%
                     width=\textwidth]{file.eps}}
\AddToShipoutPicture{
    put(0,0){\makebox(0,0)[bl]{\usebox{\mygraphic}}}
\begin{document}
  ...
\end{document}
```

上面的例子中，将 EPS 图形放大到与正文一样大小，然后存放到一个 LaTeX 盒子中。在每一页中将此盒子放置在正文的下方。

第四部分

浮动图形环境

16

浮 动 图 形 环 境

在使用字处理软件排版时，使用者可以让图形准确出现在放置的位置。但是，因为这些图形不能被分割开来，所以经常会导致糟糕的分页，将大片的空白留在页面下方。为得到专家级的排版效果，作者不得不手工调整图形的位置。这种工作是非常乏味的，尤其是几乎每次修改文档都得这样做一次。

为了既能得到专家级的排版效果，又不必手工做调整图形位置的乏味的工作，LaTeX 提供了一个浮动图形机制来自动将图形放置到合适的位置。这一机制是非常有效的。不过，它也会给那些习惯于手工调整图形的新手带来麻烦。有效的利用浮动图形机制需要注意以下几点：

- **不要使用依赖于图形放置位置的文本。** 使用如“这幅图...”或“下面的图形...”等短语要求所指的图形需在固定位置。而像“图 5...”这样的短语则允许图形出现在任意位置。
- **放松。**一些使用者在发现图形没有十分准确的出现在他们所想要的位置时，往往非常着急。这没有必要，图形的放置是 LaTeX 的工作，最好放松一些。

在接下来的几页中我们将介绍 LaTeX 是以怎样的排版规则来 决定浮动图形的位置的。为方便起见，下面列出一些最常见的关于浮动图形放置的问题。

经验总结

1. 不要束缚 LaTeX 的手脚。给出的浮动选项越多，LaTeX 做的就越好。特别地，使用选项 `[htbp]` 和 `[tbp]` 就很好。见第 16.2 节。

2. 很多人发现缺省的浮动参数过于严格了。下面的命令

```
\renewcommand{\textfraction}{0.15}
\renewcommand{\topfraction}{0.85}
\renewcommand{\bottomfraction}{0.65}
\renewcommand{\floatpagefraction}{0.60}
```

 将浮动参数重新设置为更宽松的值。详见第 17.2 节。

3. LaTeX 允许图形浮动到当前页的顶部，这样会使图形在引述它的文本前出现。不喜欢这样做的用户可以使用 flafter 宏包。无需使用特殊的命令，只要简单地调入该宏包 `\usepackage{flafter}` 即可。

4. 要确保图形的浮动不超过某一特定点，可调入 placeins 宏包，并且使用 `\FloatBarrier` 命令。见第 16.3 节。

 警告： 过多使用 `\FloatBarrier` 命令会导致浮动位置难以控制或浮动参数不正确。这两种情况都是应当尽量避免的。

§ 16.1. 创建浮动图形

可以通过把命令置于一个 `figure` 环境中来生成浮动图形。在图形环境中的所有内容都会被保持在一起，浮动到合式的位置以保证得到最好的分页结果。通过使用 `\caption` 命令来为浮动图形自动地编号并加上标题。例如，下面的命令将 EPS 图形 `graph.eps` 放到一个浮动图形中。

```
\begin{figure}
  \centering
  \includegraphics[totalheight=2in]{graph.eps}
  \caption{This is an inserted EPS graphic} \label{fig:graph}
\end{figure}

The graph in Figure~\ref{fig:graph} on Page~\pageref{fig:graph}...
```

对于图形环境，应当注意：

- `\label` 命令和 `\ref`, `\pageref` 命令配合使用，可对图形标题进行交叉引用。而 `\label` 命令必须紧接着 `\caption` 命令给出。

- 如果一图形环境中没有使用 \caption 命令，那么它将是一个没有编号的浮动图形。
- 如果一图形环境中使用了多个 \caption 命令，那么它将生成多个一起浮动的图形。这在排版并列放置的图形（见第 27 章）或像第 109 页上图 19.1–19.7 那样复杂排列的图形时是非常有用的。
- 可用命令 \listoffigures 来得到一个图形目录。
- 缺省地，图形标题将在图形目录中列出。\caption 命令有一可选项可用来将与标题文本不同的内容加到图形目录中。如：

```
\caption[List Text]{Caption Text}
```

 会在标题中使用 Caption Text 而在图形目录中使用 List Text。这在使用了特别长的标题时会很有用。
- 图形环境（figure）不能在段落中使用。因此也不能在像 parbox 或 minipage 等盒子中使用。
- 若一图形环境（figure）被置于一正文段落中，

```
....text text text text text text
\begin{figure}
   ....
\end{figure}
text text text text text text...
```

 那么它在正文段落结束之前不会被处理。

§ 16.2. 图形的放置

图形（figure）环境有一个可选参数项允许用户来指示图形有可能被放置的位置。这一可选参数项可以是下列字母的任意组合。

h **当前位置。** 将图形放置在正文文本中给出该图形环境的地方。如果本页所剩的页面不够，这一参数将不起作用。

`t` **顶部。** 将图形放置在页面的顶部。

`b` **底部。** 将图形放置在页面的底部[1]。

`p` **浮动页。** 将图形放置在一只允许有浮动对象的页面上。

注：

- 如果在图形环境中没有给出上述任一参数，则缺省为 `[tbp]`。
- 给出参数的顺序不会影响到最后的结果。因为在考虑这些参数时 LaTeX 总是尝试以 `h-t-b-p` 的顺序来确定图形的位置。所以 `[hb]` 和 `[bh]` 都使 LaTeX 以 `h-b` 的顺序来排版。
- 给出的参数越多，LaTeX 的排版结果就会越好。`[htbp]`，`[tbp]`，`[htp]`，`[tp]` 这些组合得到的效果不错。
- 只给出单个的参数项极易引发问题[2]。如果该图形不适合所指定的位置，它就会被搁置并阻碍对后面的图形的处理。一旦这些阻塞的图形数目超过了 18 幅这一 LaTeX 所能容许的最大值，就会产生 “Too Many Unprocessed Floats” 的错误（见第 16.3 节）。

参见文献
[1, 第 198 页]

当 LaTeX “试图” 放置一浮动图形时，它将遵循以下规则：

1. 图形只能置于由位置参数所确定的地点。
2. 图形的放置不能造成超过版心的错误（`overfull page`）。
3. 图形只能置于当前页或后面的页中[3]。所以图形只能 “向后浮动” 而不能 “向前浮动”。
4. 图形必须按顺序出现。这样只有当前面的图形都被放置好之后才能被放置。

[1] 当一幅图形被放置在页面的底部时，如果此页有脚注的话，它将位于所有脚注的下方。现在还没有办法来避免这种情况。

[2] 实际上，`[h]` 选项不可能单独使用。由于使用单个的 `[h]` 选项所导致的糟糕结果使得较新版本的 LaTeX 自动将其改为 `[ht]`。

[3] 因为图形可浮动到当前页的顶部，所以它可能会出现在它所在文本的前面。要防止出现这种情况，可使用 flafter 宏包。

- 只要前面有未被处理的图形，一幅图形就不会被放在当前位置。
- 一幅 “不可能放置” 的图形将阻碍它后面的图形的放置。直到文件结束或达到 LaTeX 的浮动限制。参见第 16.4 节。

同样地，一表格也只能在其前面的表格都被处理完后才能被放置。不过，表格在排版时是跳过图形而单独处理的。

5. 必须符合在第 17 章中给出的审美条件。例如，一页上的浮动对象的数目不能超过 `totalnumber`。 在浮动位置选项前加上一个惊叹号（如 `\begin{figure}[!ht]`）会使 LaTeX 忽略应用于文本页的审美条件，试图用最严格的标准来放置浮动图形。不过，`!` 不会影响应用于浮动页的审美条件。

§ 16.3. 清除未处理的浮动图形

使用浮动图形的一大优势就是 LaTeX 不需要在将它们放置在输入它们的地方。 LaTeX 会将它们暂时保存，在更合适的地点加以放置。当一浮动图形已被 LaTeX 读入，但还没有将它放到页面上时，这一图形被称为 “未处理浮动图形”。虽然 LaTeX 通常对浮动图形的处理很好，但有时还是需要强迫 LaTeX 去处理那些未处理的浮动图形。

下面的三个方法都可以用来清除未处理的浮动图形。这些命令必须分开使用。同时，过多地使用这些命令会使得你有时得自己来管理浮动图形的位置或意味着浮动图形的放置参数是错误的（见第 17 章）。

clearpage

最基本的用来清除未处理的浮动图形的方法就是使用 `\clearpage` 命令。它可让 LaTeX 排版所有未处理的浮动图形并开始一新页。尽管这一命令很有效，但它也常常导致页面的下方出现很大的空白。

FloatBarrier

对于大多数情况，最好的方法是使用 placeins 宏包提供的 `\FloatBarrier` 命令。使用 placeins 宏包的方法如下：

- \FloatBarrier 使所有未处理的浮动图形立即被处理。与 \clearpage 不同的是，它不会开始一新页。
- 如果经常要求浮动图形在它们所在的章节中排出，可在调用 placeins 宏包时使用 section 选项：

 \usepackage[section]{placeins}

 这样会重新定义 \section 命令，在每一个 section 前都加上一个 \FloatBarrier 命令。

 注意这个 [section] 选项是很严的。举例来说，如果在一页的中间开始一新的 section，那么上面这个 section 的浮动图形就不能放置在这一页的底部。
- 使用 below 选项：

 \usepackage[below]{placeins}

 会比使用 section 选项松一些。它可以允许上一个 section 的浮动图形出现在下一 section 的开始部分，只要在同一页中有上一个 section 的内容。

afterpage/clearpage

afterpage 宏包提供了命令 \afterpage，该命令将在下一自然分页时执行。因此，用

\afterpage{\clearpage}

会使所有未处理的浮动对象在下一分页前被清除完。

使用 \afterpage{\clearpage} 并不总可以解决浮动限制问题（见第 16.4 节）。因为它只是在下一页结束前才会执行 \clearpage 命令，而在下一页结束前，未处理的浮动对象可能已超过了 LaTeX 的限制。

\afterpage{\clearpage} 命令在排版较小的浮动页图形时特别有用。而命令 \floatpagefraction（见第 17.2 节）会 阻止 “太小” 的图形在浮动页上出现，由于浮动参数改变选项 ! 不会应用于浮动页，[!p] 不会破除 \floatpagefraction 的限制，使用 \afterpage{\clearpage} 是一个克服 \floatpagefraction 的限制而又不会导致有较多空白的正文页的一个简单的办法。

§ 16.4. 过多未处理的浮动对象

如果一浮动对象不能被即时处理，它就会被放到未处理的浮动对象队列中等待处理。由于在 LaTeX 中这一队列只能有 18 个位置，所以当未处理的浮动对象的数目超过这一限制时就会导致发生 “Too Many Unprocessed Floats” 的错误。造成这种错误的原因有四：

1. 最常见的原因是浮动位置选项与浮动位置参数冲突。例如一给定 `[t]` 选项的图形，如果它的高度超过了 `\topfraction` 的值，就会被放到等待处理队列中。所以给出尽可能多的浮动位置选项就会解决类似的问题。

2. 不适当的浮动式样参数值会造成一些图形无法放置。要防止出现这种情况，一定要确保所使用的浮动式样参数值满足第 17.2 节中对此的要求。

3. 在很少的情况下，如使用了很多浮动对象和边注（和浮动对象的处理机制相同），可能确实需要较大的等待队列，这时可使用 morefloats 宏包将等待队列的数目限制增加到 36。

4. 如果超过 18 幅图形在其中间没有任何文本的情况下被读入，就会超出 LaTeX 浮动放置队列的最大数目。可能的解决办法是：

 (a) 将图形散布在正文中。这会使得有足够的文本来自然分页，LaTeX 也会更容易地处理浮动对象。

 (b) 在这些图形之间加入 `\clearpage`。这样做可能得花费一些时间来调整页面以避免产生有很大空白的页（因为 `\afterpage{\clearpage}` 只在下一自然分页才会执行 `\clearpage`，而在这种情形下在下一自然分页前就会超过限制了。所以不会起作用。）。

 (c) 因为这里没有文本，所以图形也不用浮动。故最好的解决办法是采用第 20 章中的方法来构建非浮动的图形，而用 `\vspace` 和 `\vfill` 来提供竖直间距。

17

定 制 浮 动 位 置

下面列出的这些式样参数是 LaTeX 用来避免出现像一页上放置了过度的浮动对象等糟糕的情况。如果在正文中修改了这些参数的值，那么它们在下一页才会生效。如果在导言区修改了这些参数，那么会对整个文档都起作用。

§ 17.1. 浮动图形放置的计数器

表 17.1 中所给出的三个计数器可用于防止 LaTeX 将过多的浮动对象置于一文本页中，但它们不会影响浮动页。在浮动位置选项前加上 ! 会让 LaTeX 忽略这些计数器。这些计数器的值可用 \setcounter 命令来设置。例如：

```
\setcounter{totalnumber}{2}
```

会阻止 LaTeX 将多于 两个的浮动对象放置到一文本页中。

表 17.1: Float Placement Counters

topnumber	可以位于一页顶部的浮动对象的最大数目（缺省值为 2）。
bottomnumber	可以位于一页底部的浮动对象的最大数目（缺省值为 1）。
totalnumber	可以位于一页中的浮动对象的最大数目（缺省值为 3）。

§ 17.2. 图形环境中的各种比例参数

表 17.2 中给出的命令用来控制一页中有多大比例的区域可用来放置浮动对象（这里的比例是指浮动对象的高度除以正文高度 \textheight）。前面三个命令只作用于文本页，而最后一个命令只作用于浮动页。在浮动位置选项前加上 ! 会让 LaTeX 忽略前面三个命令，而 \floatpagefraction 总是起作用的。这些命令的值可以用 \renewcommand 来修改。例如：

```
\renewcommand{\textfraction}{0.3}
```

限定浮动对象不得占用文本页的 70% 以上。

表 17.2: Figure Placement Fractions

\textfraction	页面中必须用来排放文本的最小比例。缺省值为 0.2，即一页中浮动对象所占的比例不得超过 80%。
\topfraction	页面顶部可以用来放置浮动对象的高度与整个页面高度的最大比例。缺省值为 0.7，即放置在页顶部的浮动对象所占的高度不得超过整个页面高度 70%。同样地，如果多个使用了选项 t 的浮动对象的高度和超过了整个页面高度的 60%，即使它们的数目没有超过 topnumber 的值，仍将一个也不会被放置在页面顶部。
\bottomfraction	页面底部可以用来放置浮动对象的高度与整个页面高度的最大比例。缺省值为 0.3，这使得如果浮动对象的高度不超过整个页面高度的 40%，可以允许放置在页面底部。
\floatpagefraction	浮动页中必须由浮动对象占用的最小比例。因此在一浮动页中空白所占的比例不会超过 1 - \floatpagefraction。缺省值为 0.5。

各种比例的使用指引

这些比例的缺省值既 可以防止浮动对象占据过多的文本页面，也可以防止在一有很大的空白的浮动页上放置很小的图形。虽然这些缺省值让 LaTeX 工作地很好，但有时显得稍稍严了些，结果导致有些图形被浮动到距标明它们的命令很远的地方。这种情况下，可以将这些比例的值放宽松些，例如：

```
\renewcommand{\textfraction}{0.15}
\renewcommand{\topfraction}{0.85}
\renewcommand{\bottomfraction}{0.65}
\renewcommand{\floatpagefraction}{0.60}
```

在修改这些比例值的时候必须要小心，不适当的比例值会导致低劣的排版结果等问题。要避免出现这类问题，应该遵守以下的规则：

\textfraction

不要让 \textfraction 的值小于 0.15，因为这会导致令人难以阅读的文本页。如果一幅图的高度超过了 \textwidth 的 85%，那么将它单独放置到一浮动页上的效果肯定比勉强将它放置到一文本页，而且下方还有一两行文本的效果好得多。

永远不要将 \textfraction 的值设为零。这样作会让 LaTeX 感到迷惑并导致低劣的排版结果。

\topfraction

永远不要使 \topfraction 的值小于 1 - \textfraction。否则会使 LaTeX 的浮动定位算法发生冲突。

\bottomfraction

好的排版风格不提倡在页面的底部放置太多的图形，故 \bottomfraction 的值一般要比 \topfraction 的值小。永远不要使 \bottomfraction 的值为零。这样作会让 LaTeX 的浮动定位算法发生冲突。

\floatpagefraction

如果 \floatpagefraction 的值很小，那么每一浮动页上就会只放置一个浮动对象。当放置的浮动对象很小的时候，会使浮动页上出现很大面积的空白。

如果 \floatpagefraction 的值大于 \topfraction 的值，使用了 [tp] 选项的图形就有可能变成 “刺”。比如一 [tp] 图形的高要大于 \topfraction 的值，却比 \floatpagefraction 的值小，那么由于它既无法放置在文本页上，也无法放置在浮动页上，所以就成为一根 “刺”。为避免出现这样的图形， \topfraction 和 \floatpagefraction 必须满足以下的不等式：

\floatpagefraction $\leq$ \topfraction - 0.05

后面的 0.05 这一项是因为文本页和浮动页有不同的竖直间距[1]。同样地，如果使用了 [bp] 或 [hbp] 图形， \floatpagefraction 和 \bottomfraction 要满足：

\floatpagefraction ≤ \bottomfraction - 0.05

注意缺省值并不满足上面的不等式，在处理 [bp] 或 [hbp] 图形时可能会有问题。

§ 17.3. 限制浮动

\suppressfloats 阻止在当前页的顶部或底部出现浮动对象。但是不会影响图形出现在当前位置或那些在位置选项前使用了 ! 的图形。

在一幅图形后紧接着给出 \suppressfloats[t] 会阻止图形出现其在文本中的位置的上方。 flafter 宏包重定义了 LaTeX 的浮动算法来在整个文档中都会这样做。

表 17.3: Suppress oats Options

\suppressfloats[t]	限定在当前页的顶部没有其它的浮动对象。
\suppressfloats[b]	限定在当前页的底部没有其它的浮动对象。
\suppressfloats	限定在当前页的顶部和底部都不能出现其它的浮动对象。

[1]特别地，在比较图形的高度所占比例和 \topfraction 时， \textfloatsep 和其它文本页浮动间距都被计算在内。而对浮动页来说，在测试图形的高度所占比例是否超过了 \floatpagefraction 时，浮动间距是不被计算在内的。所以必须从 \topfraction 中减去 \textfloatsep 除以 \textheight 的值。详见第 18.1 节。

18

定 制 图 形 环 境

§ 18.1. 图形的间距

表 18.1 中给出的长度控制着两幅图形之间或图形与正文之间的间距。与其它的 LaTeX 长度不同的是，这三个都是橡皮长度，这就使得它们可以缩短或拉长来更好的排版页面。这些长度可用 \setlength 命令来设定。例如：

```
\setlength{\floatsep}{10pt plus 3pt minus 2pt}
```

将正常的 \floatsep 的值设定为 10pt。并且在需要时可缩短到 8pt 或拉长到 13pt。

表 18.1 中给出的长度不会影响浮动页上各浮动对象之间的距离。它们由表 18.2 中给出的长度控制。单位 fil 允许无限伸长，就像由 \vfill 产生的垂直距离一样。当在一段距离中出现多个 fil 时，它们将按比例充满这段距离。

表 18.1: Figure Spacing for Text Pages

\floatsep	出现在页面的顶部或底部的浮动对象之间的垂直距离。缺省为 12pt plus 2pt minus 2pt。
\textfloatsep	出现在页面的顶部或底部的浮动对象与文本之间的垂直距离。缺省为 20pt plus 2pt minus 4pt。
\intextsep	出现在页面中间的浮动对象（如使用了 h 选项的浮动对象）与上下方文本之间的垂直距离。缺省为 12pt plus 2pt minus 2pt。

表 18.2: Figure Spacing for Floatpages

\@fptop	浮动页中顶部的浮动对象上方的空白。缺省为 0pt plus 1.0fil。
\@fpsep	浮动页中的浮动对象之间的距离。缺省为 8pt plus 2.0fil。
\@fpbot	浮动页中底部的浮动对象下方的空白。缺省为 0pt plus 1.0fil。

表 18.3: Figure Rule Commands

\topfigrule	在一页顶部的最后一个浮动对象后，\textfloatsep 前被执行的命令（见第 18.1 节）。
\bottomfigrule	在一页底部第一个浮动对象前，\textfloatsep 后被执行的命令。

在表 18.2 中的长度名字前的 @ 表示这是一个 LaTeX 内部命令[1]。所以，所有改变这些长度的 \setlength 命令都必须放到 \makeatletter 和 \makeatother 之间。例如：

```
\makeatletter
  \addtolength{\@fpsep}{4pt}
\makeatother
```

将浮动页中浮动对象之间的距离增加了 4pt。

§ 18.2. 图形上下方的水平线

通过重新定义 \topfigurerule 和 \bottomfigurerule 可在页面顶部或底部的文本和图形之间画上一水平线。尽管 \topfigurerule 和 \bottomfigurerule 是已经定义的 LaTeX 命令，但是它们奇特的定义方式要求在重定义时用 \newcommand 而不是 \renewcommand。

为了不破坏版面，这些命令所加的标尺的高度必须为零。例如要划一条 0.4pt 的水平线，就必须加上一 -0.4pt 的距离：

[1] 为实现它的功能，LaTeX 使用了很多普通用户无需涉及的内部命令。为防止这些内部命令名字和用户定义的命令的名字发生冲突，，LaTeX 在它的内部命令名字前加上了一个 @。由于 LaTeX 命令的名字只能包含字母，所以通常不可能定义一个含有 @ 的命令。不过，命令 \makeatletter 让 LaTeX 把 @ 当作字母，从而可以定义带有 @ 的命令。命令 \makeatother 则重新令 LaTeX 不把 @ 当作字母。用户所有涉及到 LaTeX 内部命令的代码都必须包含在 \makeatletter 和 \makeatother 之间。

```
\newcommand{\topfigrule}{\hrule\vspace{-0.4pt}}
```

因为 \topfigrule 在 \textfloatsep 之前被执行，上面的命令没有在图形与水平线之间留出距离。下面的命令则在图形与水平线之间留出了 5pt 的空间。

```
\newcommand{\topfigrule}{%
  \vspace*{5pt}\hrule\vspace{-5.4pt}}
\newcommand{\botfigrule}{%
  \vspace*{-5.4pt}\hrule\vspace{5pt}}
```

在这里 \topfigrule 的定义中，首先向下移动 5pt（进入到 \textfloatsep 的区域）来给出图形与水平线之间的距离，然后画上一高为 0.4pt 的水平线，最后再向上移动 5.4pt 以补偿前面向下的位移。同样地， \botfigrule 在图形与水平线之间留出了 5pt 的空间。

由于上面的命令使得图形与水平线之间的距离为 5pt ，所以水平线与文本之间的距离为 \textfloatsep - 5pt （见第 18.1 节）。

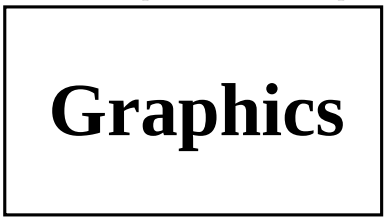
水平线的线宽缺省为 0.4pt，并可用 \hrule 命令的 height 选项来改变。

```
\newcommand{\topfigrule}{%
  \vspace*{5pt}{\hrule height0.8pt}\vspace{-5.8pt}}
\newcommand{\botfigrule}{%
  \vspace*{-5.8pt}{\hrule height0.8pt}\vspace{5pt}}
```

需要注意下面几点：

- \topfigrule 和 \botfigrule 命令对浮动页上的图形和放置在当前位置的图形（如使用了 h 选项）不起作用。如果一放置在当前位置的图形正好位于页面的顶部或底部，也不会画上水平线。
- 水平线的长度与文本的宽度相等。而不管图形是不是很宽。
- 因为 LaTeX 的 \rule 命令在 \parskip 不为零时会产生额外的空白，所以代之以 TeX 命令 \hrule。

图 18.1: Caption Above Graphic



§ 18.3. 图形与标题的间距

LaTeX 假定图形的标题位于图形的下方，故而在标题上方保留了更多的空白。因此

```
\begin{figure}
\centering
\caption{Caption Above Graphic}
\includegraphics[width=2in]{graphic.eps}
\end{figure}
```

生成的图 18.1 中标题和图形非常接近。

标题上下方的间距由长度 \abovecaptionskip 和 \belowcaptionskip （缺省分别为 10pt 与零）。可以用标准的 LaTeX 命令 \setlength 和 \addtolength 来修改这些长度。例如：

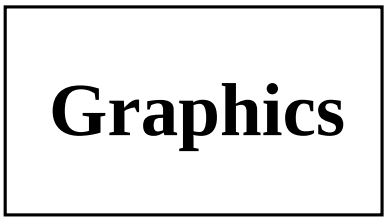
```
\begin{figure}
\setlength{\abovecaptionskip}{0pt}
\setlength{\belowcaptionskip}{10pt}
\centering
\caption{Caption Above Graphic}
\includegraphics[width=2in]{graphic.eps}
\end{figure}
```

得到图 18.2。其中标题的上方没有额外的空白，与图形之间则有 10pt 的距离。

如果一个文档的所有浮动对象的标题都位于该对象的上方，那么可将命令

```
\setlength{\abovecaptionskip}{0pt}
\setlength{\belowcaptionskip}{10pt}
```

图 18.2: Caption Above Graphic



放到导言区里，从而对整个文档都起作用。如果只是有一部分标题要求位于浮动对象的上方，那么可定义如下的命令：

```
\newcommand{\topcaption}{%
  \setlength{\abovecaptionskip}{0pt}%
  \setlength{\belowcaptionskip}{10pt}%
  \caption}
```

在希望得到上方标题的时候可用 \topcaption{标题文本} 来代替 \caption{标题文本} 即可。

§ 18.4. 标题的标记

缺省情况下，LaTeX 会在图形的标题开头加上像 “Figure 13: ” 这样的标记。其中的 “Figure” 可以通过重定义 \figurename 来更改。例如，下面的命令

```
\begin{figure}
  \centering
  \includegraphics[width=2in]{graphic.eps}
  \renewcommand{\figurename}{Fig.}
  \caption{This is the Caption}
\end{figure}
```

得到如图 18.3 的结果。至于标题文本的字体，分隔符 “:” 以及其它标题属性的修改可用 caption2 宏包（见第 19 章）来完成。

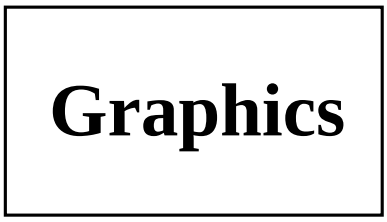


Fig. 18.3: This is the Caption

§ 18.5. 将图形放于文档的最后

有些期刊要求将图表和正文文本分开排放。这时可用 endfloat 宏包，它可以将浮动对象放置到文档的最后。使用 endfloat 时可用

```
\usepackage{endfloat}
```

将它调入。另外，这个宏包在用 \usepackage 调入时还有一些选项，包括：

- 在邻近浮动对象的文本中会放置像 “[Figure 4 about here.]” 之类的说明。要取消这一功能可在调入宏包时使用 nomarkers 选项。

  ```
  \usepackage[nomarkers]{endfloat}
  ```

 另外，说明中的文本可通过重定义命令 \figureplace 和 \tableplace 来更改。例如：

  ```
  \renewcommand{\figureplace}{%
    \begin{center}%
    [\figurename~\thepostfig\ would appear here.]%
  \end{center}
  ```

- 在图形和表格之前会有一列表。可使用 nofiglist 和 notablist 宏包选项来取消这一功能。

- fighead 和 tabhead 宏包选项分别在图形和表格前加上章节标题。

- 图形放置在表格之前，也可用 tablefirst 宏包选项来改变这一顺序。

- 在每一图形和表格后会执行 \clearpage 命令，从而使得每一页只有一个浮动对象。这可通过修改 \efloatseparator 来改变。例如，

```
\renewcommand{\efloatseparator}{\mbox{}}
```

 会在每一浮动对象后面放置一个空的盒子。

19

使 用 caption2 宏 包 来 定 制 标 题

第 18.3 节和第 18.5 节分别介绍了如何定制浮动图形的标题之标记和标题上下方的空白。对于标题的其它属性的自由控制，则利用 caption2 宏包[1]来完成。

caption2 宏包可以和很多与浮动对象有关的宏包一起使用。它正式声称支持 float, longtable, subfigure，不过实际上也和 floatfig, rotating, supertabular, wrapfig 等在一起工作的很好。

用法：
```
\usepackage[选项]{caption2}
```

这里*选项* 的具体说明见表 19.1。

§ 19.1. 标题式样

图 19.1–19.7 展示了 caption2 宏包定义的下列标题式样。

normal 标题文本两边对齐，其中最后一行为左对齐。

center 标题文本居中。

flushleft 标题文本左对齐。

flushright 标题文本右对齐。

[1]由于最早的 caption 宏包有很多副作用（比如要求在其它宏包后被调入后再才能被调入），所以被完全重新写过，命名为 caption2。尽管从技术角度来说 caption2 仍是 beta 版，但在使用中却也是非常稳定有效的。

表 19.1: caption2 选项

标题式样	normal, center, flushleft, flushright, centerlast, hang, indent	选择标题的式样（详见第 19.1 节）。
标题字号	scriptsize, footnotesize, small, normalsize, large, Large	选择标题的标记和文本的字体大小。
标记字形	up, it, sl, sc	选择标题的标记的字形，不会影响到标题的文本。
字体序列	mb, bf	选择标题的标记的字体序列，即字体的宽度或权重。不会影响到标题的文本。
标记字族	sl, sf, tt	选择标题的标记的字族，可为 Roman, San Serif 或 Typewriter 字体。不会影响到标题的文本。
单行标题	oneline, nooneline	控制是否采用单行标题格式（见第 19.3 节）。

centerlast 标题文本两边对齐，其中最后一行居中。

indent 与 **normal** 式样相似，只是标题文本从第二行开始，每行行首缩进由命令 \captionindent 给出的长度。因为 \captionindent 的缺省值为零，通常用像 \setlength{\captionindent}{1cm} 这样的命令来设置缩进值。

hang 与 **normal** 式样相似，只是标题文本从第二行开始，每行行首缩进与标题标记宽度相等的长度。

通常这些标题式样在调入宏包时给出，如：

```
\usepackage[centerlast]{caption}
```

将使整个文档中的标题都为 **centerlast** 式样。

§ 19.2. 标题式样的变换

\captionstyle 命令用来改变标题的式样。将这一命令置于一环境中时，仅仅改变这一环境中的标题式样。例如：

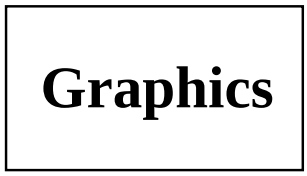


图 19.1: Normal Caption Style Normal Caption Style Normal Caption Style Normal Caption Style

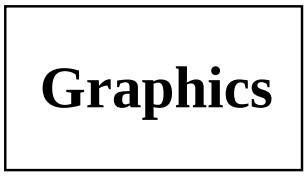


图 19.2: Center Caption Style Center Caption Style Center Caption Style Center Caption Style

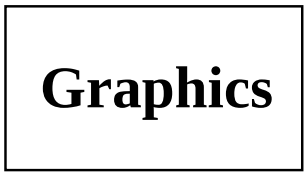


图 19.3: Centerlast Caption Style Centerlast Caption Style Centerlast Caption Style Centerlast Caption Style

```
\begin{figure}
  \captionstyle{centerlast}
  \centering
  \includegraphics[width=3in]{graphic.eps}
  \caption{Centerlast Caption Style. Centerlast Caption Style.}
  \end{figure}
```

只改变这一幅图形的标题式样。因为 `\captionstyle` 命令是置于一个浮动图形环境中的。而

```
\captionstyle{centerlast}
\begin{figure}
 \centering
  \includegraphics[width=3in]{graphic.eps}
  \caption{Centerlast Caption Style. Centerlast Caption Style.}
  \end{figure}
```

将此图与以后的图形的标题式样都改为 **centerlast**。因为命令 `\captionstyle` 是置于浮动图形环境外的。

§ 19.3. 单行标题

如果标题只有一行，上节介绍的所有的式样都会居中放置这一标题。为在标题文本只有一行的情况下，仍然可以应用这些不同的式样，必须在调入 caption2 时给出 `nooneline` 选项。如

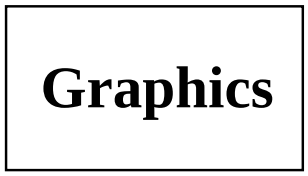


图 19.4: Flushleft Caption Style Flushleft Caption Style Flushleft Caption Style Flushleft Caption Style

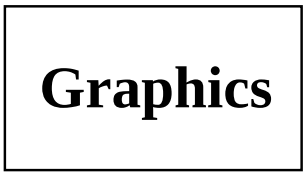


图 19.5: Flushright Caption Style Flushright Caption Style Flushright Caption Style Flushright Caption Style

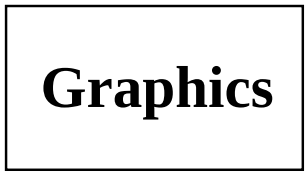


图 19.6: Indent Caption Style Indent Caption Style Indent Caption Style Indent Caption Style

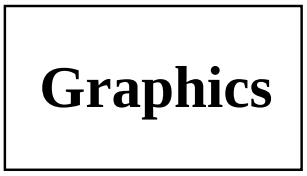


图 19.7: Hang Caption Style Hang Caption Style Hang Caption Style Hang Caption Style

```
\usepackage[nooneline,flushleft]{caption2}
```

使得所有的标题文本（包括单行标题）都采用 `flushleft` 式样。若想在文本中改变 `nooneline` 选项，可使用命令 `\onelinecaptiontrue` 来居中放置单行标题，而命令 `\onelinecaptionfalse` 使得重新对单行标题应用所选择的标题式样。例如：

```
\begin{figure}
  \captionstyle{flushleft}
  \onelinecaptionstrue
  \centering
  \includegraphics[width=2.5in]{graphic.eps}
  \caption{First Caption}
\end{figure}
```

如同图 19.8 所示，标题被居中放置。而下面的命令：

```
\begin{figure}
  \captionstyle{flushleft}
  \onelinecaptionsfalse
```

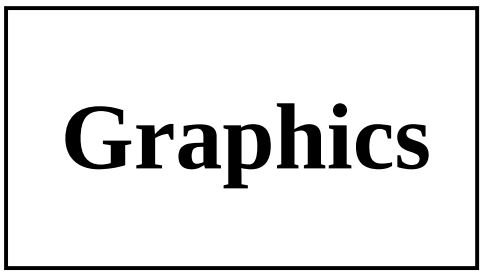


图 19.8: First Caption

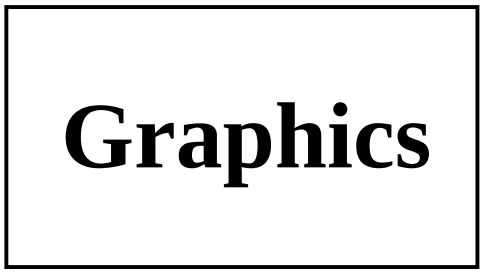


图 19.9: Second Caption

```
    \centering
    \includegraphics[width=2.5in]{graphic.eps}
    \caption{Second Caption}
  \end{figure}
```

使得单行标题如图 19.9 所示，采用左对齐式样。

§ 19.4. 标题的宽度

caption2 宏包提供了直接指定标题的宽度及其两边的空白的功能。

- `\setcaptionwidth{width}` 设定标题的宽度为 `width`，这里的 `width` 为任意有效的 TEX 度量单位。
- `\setcaptionmargin{mar}` 设定标题任一边的空白为 `mar`，从而使得标题的宽度为标准宽度减去两倍的 `mar`。

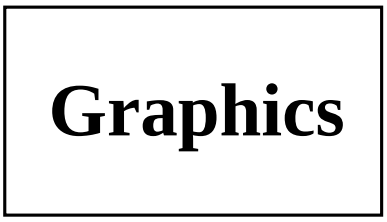


图 19.10: Figure Caption Limited to Two Inches

如果 mar 为一负值，那么标题的宽度要比标准的宽度宽一些。这在子图和小页环境中非常有用。

例如，命令

```
\begin{figure}
  \setcaptionwidth{2in}
  \centering
  \includegraphics[width=2in]{graphic.eps}
  \caption{Figure Caption Limited to Two Inches}
\end{figure}
```

使得标题的宽度为 2 英寸，结果如图 19.10。

上面的例子直接设定了标题的宽度。还有一种方法是通过给定标题和两边页边界的距离来间接设定标题的宽度。例如，命令

```
\begin{figure}
  \setcaptionmargin{1in}
  \centering
  \includegraphics[width=2in]{graphic.eps}
  \caption{Figure Caption Where There is One Inch of Spacing
           between the Caption and Each Margin}
\end{figure}
```

使得标题到两边页边界的距离为 1 英寸。如图 19.11 所示。

下面主要介绍一下如何将标题的宽度设为图形的宽度。如果图形的宽度已知，这将是非常容易的。

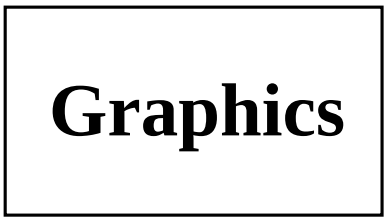


图 19.11: Figure Caption Where There is One Inch of Spacing between the Caption and Each Margin

```
\includegraphics[width=3in]{file.eps}
\setcaptionwidth{3in}
\caption{...}
```

如果图形的宽度未知，可以通过将它放到一个盒子里然后测量盒子的宽度来得到。

```
\newsavebox{\mybox}
\newlength{\mylength}
...
\begin{figure}
  \centering
  \sbox{\mybox}{\includegraphics[height=3in]{file.eps}}
  \settowidth{\mylength}{\usebox{\mybox}}
  \setcaptionwidth{\mylength}
  \usebox{\mybox}
  \caption{This is a figure with a very, very, very,
           very, very, very, very long caption}
\end{figure}
```

这种方法也可应用于表格。 \mybox 和 \mylength 可在文档中使用多次，而 \newlength 和 \newsavebox 只须声明一次即可。

§ 19.5. 标题的分隔符

在标题中，缺省的分隔符 ":" 可通过重定义 \captionlabeldelim 来加以改变。例如，

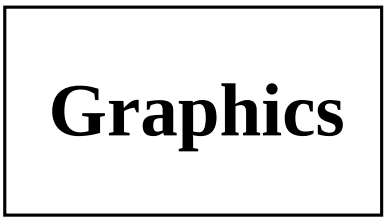


图 19.12. Caption with New Delimiter

```
\begin{figure}
  \renewcommand{\captionlabeldelim}{.}
  \centering
  \includegraphics[width=2in]{graphic.eps}
  \caption{Caption with New Delimiter}
\end{figure}
```

将图 19.12 中的分隔符改为句点 “.”。如果希望在句点后面加上一点距离，可用下面的命令来得到。

```
\renewcommand{\captionlabeldelim}{.~}
```

§ 19.6. 标题的字体

当在 \usepackage{caption2} 中使用 scriptsize,...,Large 选项时，标题的标记和文本的字号均会相应的改变。而使用 up，it，sl，sc，md，bf，rm，sf，tt 选项时只作用于标题标记。

caption2 宏包也允许用户设定单独的标题字体。\captionfont 命令可用来设定标题的字体（包括标记和文本），而命令 \captionlabelfont 则只设定标题标记的字体。因此若只想设定标题文本的字体，必须使用 \captionfont 来设定标题文本的字体，同时用 \captionlabelfont 来设定标题标记的字体，包括取消一些由 \captionfont 设置的字体属性。下面的命令可以有效的生成标题：

```
{\captionfont%
  {\captionlabelfont \captionlabel \captionlabeldelim}%
   \captiontext}
```

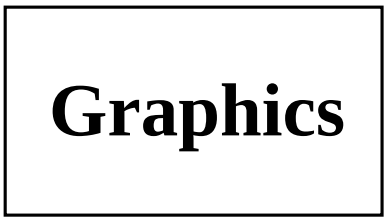


图 19.13: Test Caption

这里的 \captionlabel 命令生成标题标记，如 “**图** 1”。 \captionlabeldelim 生成标记与文本之间的分隔符 “:”。 \captiontext 则给出标题文本。

LaTeX 的字体可用字号和三个式样：字形，字族和字体序列（见 [1, 第 37,115 页]， [3, 第 170-171 页]）来描述。所有这四个字体特性均可用 \captionfont 和 \captionlabel 来指定。例如：

```
\begin{figure}
  \renewcommand{\captionfont}{\Large \bfseries \sffamily}
  \renewcommand{\captionlabelfont}{}
  \centering
  \includegraphics[width=2in]{graphic.eps}
  \caption{Test Caption}
  \end{figure}
```

结果如图 19.13 所示。在这个例子中， \captionlabelfont 没有是空的，这意味着它没有改变标题缺省的字体属性和由命令 \captionfont 设定的标题标记的字体属性。由于没有给出字形，所以整个标题的字形为缺省的 upright 字体。

图 19.14 由下面的命令得到：

```
\begin{figure}
  \renewcommand{\captionfont}{\Large \bfseries \sffamily}
  \renewcommand{\captionlabelfont}{\small}
  \centering
  \includegraphics[width=2in]{graphic.eps}
  \caption{Test Caption}
\end{figure}
```

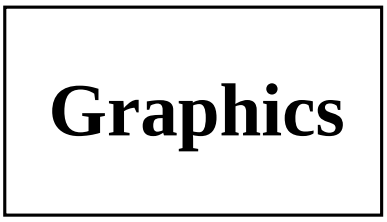


图 19.14: **Test Caption**

在这个例子中，由 \captionlabelfont 给出的 \small 覆盖了由 \captionfont 指定的 \Large 字号。不过，由于 \captionlabelfont 没有指定字体序列和字族，所以 \bfseries 和 \sffamily 也应用于标题标记。

§ 19.7. 定制标题式样

caption2 宏包也允许用户定义自己的标题式样。例如下面的命令

```
\newcaptionstyle{one}{%
  \usecaptionmargin\captionfont%
  \onelinecaption%
  {{\bfseries\captionlabelfont\captionlabel\captionlabeldelim}
    \captiontext}%
  {{\centering\bfseries\captionlabelfont\captionlabel\par}%
    \captiontext}}

\newcaptionstyle{two}{%
  \usecaptionmargin\captionfont%
  {\centering\bfseries\captionlabelfont\captionlabel\par}
   \onelinecaption{\captiontext}{\captiontext}}
```

定义了标题式样 one 和 two。对于多于一行的标题，这两种式样都使用加黑的标题标记（如 **Figure 12**）并单独占据一行。而对于单行标题，式样 two 使用加黑的标题标记并单独占据一行，标题文本另起一行。式样 one 则将标题标记和文本放置在同一行，中间用分隔符隔开。下面的图 19.15 和图 19.16 是由下面的命令得到的并分别使用了上面自定义的两种标题式样。

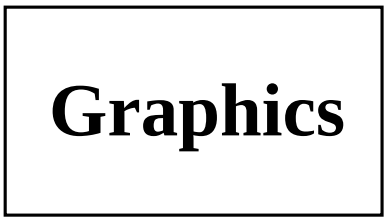


图 19.15: First Custom Caption Style

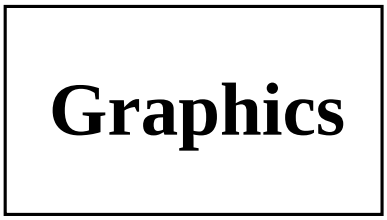


图 19.16
Second Custom Caption Style

```
\begin{figure}
  \captionstyle{one}
  \centering
  \includegraphics[width=2in]{graphic.eps}
  \caption{First Custom Caption Style}
\end{figure}

\begin{figure}
  \captionstyle{two}
  \centering
  \includegraphics[width=2in]{graphic.eps}
  \caption{Second Custom Caption Style}
\end{figure}
```

对于自定义标题式样，需要注意以下几点：

- 命令 \onelinecommand 带有两个参数：第一个在标题为单行时使用，第二个则是在标题文本多于一行时使用。
- 自定义标题式样时，不要求必须用 \captionfont 和 \captionlabelfont。不过，鼓励使用这些命令以使得所定义的式样更具灵活性。

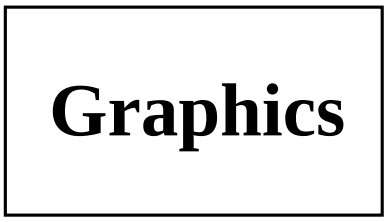


图 19.17: First Line of Caption
Second Line of Caption

例如，在上面自定义的式样中，可用 \captionlabelfont 来改变缺省的 \bfseries。如果不需要这种灵活性，那么上面自定义的标题式样的代码可以更简洁些。

§ 19.8. 标题中的断行

如果标题的文本多于一行，可用 \protect\\ 来断行。然而，当标题文本的长度不超过一行时，它们被放在一个水平盒子中来处理，所有的 \\ 或 \par 都将被忽略。

caption2 宏包允许标题文本以指定的任意长度断行。例如命令

```
\begin{figure}
  \centering
  \includegraphics[width=3in]{graphic.eps}
  \captionstyle{center}
  \onelinecaptionsfalse
  \caption{First Line of Caption \protect\\
           Second Line of Caption}
    \label{fig:caption:linebreak}
\end{figure}
```

得到图 19.17 中的标题。因为 \\ 是脆弱的[2]，必须在其前面加上 \protect。

使用 \onelinecaptionfalse 命令（或 nooneline 宏包选项）防止 LaTeX 将标题置于一水平盒子中处理以致不能断行。

[2] 一些命令（如 \textbf）不在辅助文件中储存任何信息。那些把信息储存起来以备将来使用的命令被称作具有移动参数的命令。在移动参数中使用时会崩溃的命令就被称为是脆弱的，相反则称为健壮的。

§ 19.9. 调整标题中的行距

若在文档中使用两倍行距，要在导言区中加入[3]：

```
\linespread{1.6}
```

或等价地

```
\renewcommand{\baselinestretch}{1.6}
```

这时，除了使得正文中行距为缺省值的两倍外，脚注和浮动对象中标题的行距也扩大为原来的两倍。要想在正文中使用两倍行距，而在标题中使用单倍行距，可由 setspace 宏包[4]来完成这一任务。

```
\usepackage{setspace}
\linestretch{1.6}
```

\linestrech 的值为 1 时为单倍行距， 1.2 时是一倍半行距，而为 1.6 时是双倍行距。

无论 setspace 使用与否， \captionfont 命令都可以用来调节标题文本的行距。例如：

```
\renewcommand{\captionfont}{\linespread{1.6}\normalsize}
```

使得无论正文中的行距是多少，标题标题文本为双倍行距。

[3]这样的命令也可在正文中使用，尽管这种方式被认为是蹩脚的，但也是可以用来在正文中改变行距。当使用这种方式时，必须在其后声明像 \normalsize 等字号命令以使所做的行距变化生效。

[4]尽管 doublespace 宏包也可以用来改变行距，但它并没有很好地按照 LaTeX 2ε 的标准来写，经常与其它的 LaTeX 2ε 宏包冲突，所以最好还是用 setspace。

20

不浮动的图形

如同第 16 章所介绍的那样，LaTeX 允许图形和表格“浮动”以增强排版效果。不过，偶尔也会希望一幅图形不要浮动，就放置在与它在 LaTeX 源文件中相同的位置[1]。\caption 命令可以在 figure 和 table 环境中使用是因为这两个环境各自定义了内部命令 \@captype。这样，通过定义 \@captype 就可以在 figure 和 table 环境外使用 \caption 命令。当然这时 \@captype 必须用 \makeatletter-\makeatother 命令对包围起来，使得可以在命令名中使用 @。在每次使用时可用如下的命令：

```
\includegraphics{file.eps}
\makeatletter\def\@captype{figure}\makeatother
\caption{This is the caption}
```

在导言区中定义下面的命令会更加方便。

```
\makeatletter
  \newcommand\figcaption{\def\@captype{figure}\caption}
  \newcommand\tabcaption{\def\@captype{table}\caption}
\makeatother
```

这样在正文中无论是否在图形环境中，都可用 \figcaption 来得到图形标题。同样地，无论是否在表格环境中，都可用 \tabcaption 来得到表格标题。下面

[1]因为经常会导致出现大面积的空白，不让图形浮动被认为是一种糟糕的排版风格。代之以使用 [!ht] 选项的图形环境通常会得到较好的结果。

的命令

```
This is the text before the figure.
\\[\intextsep]
  \begin{minipage}{\textwidth}
    \centering
    \includegraphics[width=2in]{graphic.eps}%
    \figcaption{This is a non-floating figure}
    \label{fig:non:float}
  \end{minipage}
\\[\intextsep]
This is the text after the figure.
```

可得到一幅不浮动的图形。对于不浮动的图形，需要注意下面几点：

- 需要使用小页环境（`minipage`）来防止在图形中出现分页的情况。
- 命令 `\\[\intextsep]` 开始一新行并在图形的前后加上垂直的空白。任意大小的空白都可以，`\intextsep`（见第 18.1 节）被用来使不浮动的图形具有与浮动图形相同的上下间距。
- 一般地，浮动图形是按照它们在 LaTeX 源文件中的顺序一一被放置的。而不浮动的图形是被立即放置到页面上，所以可能会出现图形顺序错误的情况，图形出现的顺序被打乱[2]。要避免这种顺序错乱，可在不浮动的图形前用 `\clearpage` 或 `\FloatBarrier` 命令清除未处理的浮动图形（见第 16.3 节）。
- `\figcaption` 和 `\tabcaption` 在生成边注图形（见第 21 章）以及与图形并列的表格（见第 29 章）时会很有用。

§ 20.1. float 宏包中的 [H] 位置选项

float 宏包[3] 为 `figure` 环境加上了一个 `[H]` 位置选项，从而使得用 `figure` 环境可以生成不浮动的图形。为使用此功能，须在导言区使用

[2]在这种情况下，图形目录中图形的顺序是按照图形出现的顺序，而不是图形编号的顺序。
[3]float 宏包允许用户新的浮动对象，如 `Program`，`Algorithm` 等。也可以定制加框的和加线条的浮动式样。

```
\usepackage{float}
```

并且在使用 \begin{figure}[H] 命令前声明 \restylefloat 命令（见 [3, 第 149 页]）。不过，使用 float 宏包提供的 [H] 选项会伴有下面的副作用：

1. 如果当前页没有足够的空间放置一幅使用了 [H] 位置选项的图形，该图形会被置于下一页的顶部。然而，如果当前页中有脚注的话，它将会紧接在文本后排出，而不是像通常那样置于页面的底部。这时用户必须在图形前面加上足够的空白以保证将脚注移到页面的底部。

2. 使用由 float 宏包定义的图形环境总是将标题置于图形的下方。对于一般的图形来说不会有什么影响。但是，它会影响如第 103 页上图 18.2 那样标题在上方的图形，第 138 页上图 24.1 那样标题在旁边的图形或其它比较复杂的图形排放（如第 109-110 页上的图 19.1- 19.7）。

综上所述，使用本章前面所介绍的通过定义 \figcaption 来得到不浮动的图形要比使用 float 宏包的 [H] 位置选项更好些。

21

边 注 图 形

\marginpar 命令可以用来生成边注。除非使用了 \reversemarginpar 命令，边注一般放在页面的右边（在 twoside 格式的文档中放在页面的外侧）。边注的宽度由长度 \marginparwidth 控制，而与正文之间的水平距离由 \marginparsep 决定。

边注的第一行与包含它的正文文本的那一行对齐（边注的第一行的参考点与当前基线对齐）。

边注不能分页，辱国一个边注太靠近页面的底部而无法排下时，它会在页面的底边继续排出。如果前面一个边注干扰了后面的边注，那么 LaTeX 会把后面的边注向下移动，但不会移到下一页。所以在最后完成排版前可能要调整一下边注的位置以防它离分页的地方太近。

由于 figure 环境不能在边注中使用，所以无法直接得到浮动的边注图形。这时，可以用第 20 章前面介绍的通过定义 \figcaption 来构造非浮动的边注图形。 例如，图 21.1 就由下面的命令来得到：

```
...~构造非浮动的边注图形。
\marginpar{\centering
  \includegraphics[width=\marginparwidth]{graphic.eps}%
  \figcaption{This is a Marginal Figure}
  \label{fig:marginal:fig} }
例如，图~\ref{fig:marginal:fig}~就由下面的命令来得到：
```


图 21.1: This is a Marginal Figure

图 21.1 的基线与与包含 \marginpar 的正文文本的那一行对齐。对于使用边注图形，需要注意的是：

- 由于边注图形都比较窄小，所以使用 caption2 宏包的标题式样 `flushleft` 或 `flushright` 可能会得到更好的效果。此外， caption2 宏包的命令

  ```
  \renewcommand{captionfont}{\small}
  ```

 可使标题的字体变小。详见第 19 章。

- 如同第 20 章所介绍的非浮动图形一样，边注图形会在未处理的浮动图形前排出。因此，如果希望图形按顺序出现的话，必须在边注之前使用 \clearpage 或 \FloatBarrier 命令。

- 边注的处理机制和浮动图表的处理机制是一样的，所以如果使用了太多的浮动图表和边注，就可能超出 LaTeX 所允许的未处理的浮动对象的数目。这时使用 morefloat 宏包是一种解决办法。具体见第 16.4 节。

22

宽 图 形 的 处 理

排版的易读性规则限制了一行文本中的字符个数，如果不是使用大字体或双列版式，就会使得页面的边空很大。在第 21 章中展示了边空可以用来放置边注图形。另外也可以用来得到扩展到一边或两边边空的宽图形。这可通过在浮动图形环境中嵌套一个很宽的列表环境来实现。例如，可以在导言区加入下列代码来定义一个 narrow 环境：

```
\newenvironment{narrow}[2]{%
  \begin{list}{}{%
  \setlength{\topsep}{0pt}%
  \setlength{\leftmargin}{#1}%
  \setlength{\rightmargin}{#2}%
  \setlength{\listparindent}{\parindent}%
  \setlength{\itemindent}{\parindent}%
  \setlength{\parsep}{\parskip}}%
\item[]}{\end{list}}
```

那么，所有位于 \begin{narrow}{1in}{2in} 和 \end{narrow} 之间的文本都被向左缩进 1 英寸，向右缩进 2 英寸。当使用负长度时，文本就会延伸到边空上去。

§ 22.1. 单面版式中的宽图形

在使用单面版式排版时，页面左右的边空不会因奇偶页而取不同的值，故可以不用考虑图形浮动到奇数页或偶数页的问题。下面的命令利用前面定义的 narrow 环境使得图形左边延伸到左边空中 1 英寸。

```
\begin{figure}
  \begin{narrow}{-1in}{0in}
    \includegraphics[width=\linewidth]{wide.eps}
    \caption{This is a wide figure}
  \end{narrow}
\end{figure}
```

这里给定宽度参数为 \linewidth 使得图形的宽度和 narrow 环境的宽度相等。若给定宽度参数为 \textwidth 会使图形的宽度和原来的正文宽度一样。

当使用边注时，可能希望宽图形精确延伸到边注的边界（使得图形的宽度为 \textwidth+\marginparwidth+\marginparsep）。这时，可以定义一新长度 \marginwidth 并将它设为 \marginparwidth+\marginparsep。例如：

```
\newlength{\marginwidth}
  \setlength{\marginwidth}{\marginparwidth}
  \addtolength{\marginwidth}{\marginparsep}
```

接着在 \begin{narrow} 中使用 -\marginwidth 来达到目的。

§ 22.2. 双面版式中的宽图形

在使用双面版式排版时，页面左右的边空因奇偶页而取不同的值，且使用宽图形时常常希望图形延伸到装订的那一边（奇数页的左边，偶数页的右边）。在这种情形下，需要使用 ifthen 宏包提供的 \ifthenelse 命令来根据图形出现在奇数页或偶数页而使用不同的命令。例如：

```
\usepackage{ifthen}
...
\newlength{\marginwidth}
```

```
  \setlength{\marginwidth}{\marginparwidth}
  \addtolength{\marginwidth}{\marginparsep}
\begin{figure}
  \ifthenelse{\isodd{\pageref{fig:wide}}}%
    {% BEGIN ODD-PAGE FIGURE
    \begin{narrow}{0in}{-\marginwidth}
    \includegraphics[width=\linewidth]{wide.eps}
    \caption{Figure Caption}
    \label{fig:wide}
    \end{narrow}
    }% END ODD-PAGE FIGURE
    {% BEGIN EVEN-PAGE FIGURE
    \begin{narrow}{-\marginwidth}{0in}
    \includegraphics[width=\linewidth]{wide.eps}
    \caption{Figure Caption}
    \label{fig:wide}
    \end{narrow}
    }% END EVEN-PAGE FIGURE
\end{figure}
```

结果如图 22.1。由于 \ifthenelse 使用命令 \pageref 作为输入，所以需要 LaTeX 运行足够的次数后才能正确地排出。

注：如果使用了 hyperref 宏包，上例中的 \pageref 应替换为 \hypergetpageref 。



图 22.1: Figure Caption

23

横 排 的 图 形

在一竖排版的文档中，有三种方法来得到横排的图形。

1. lscape 宏包提供了一个 `landscape` 环境，将纸张的左边界作为页面的顶部，使得在此环境中的文本，表格和图形都被横排。
2. rotating 宏包提供了一个 `\sidewaysfigure` 环境，与 `figure` 环境相似，只是其中的图形被横排。
3. rotating 宏包提供了一个 `\rotcaption` 命令，与 `\caption` 命令相似，只是标题被横排。

以上三中方法的区别：

- 方法 1 和 2 将横排的图形放到单独的一页上，而方法 3 则生成一个并不需要单独一页来放置的浮动对象。
- 方法 2 只是将其中的图形横排，而方法 1 则将位于 `landscape` 环境中的任何文本，图形和表格都横排在一页中。 `landscape` 环境具有分页的能力，可连续生成多个横排页面[1]。
- 使用方法 2 得到的整页的图形可以浮动以求得最佳排版效果，而方法 1 得到的图形是不能浮动的[2]。

[1] `landscape` 环境能很好的与 longtable 宏包配合，从而得到连续多页横排的超长表格。
[2] 在 `landscape` 环境中声明的浮动图形只能在横排页中浮动。

- 因为方法 1 和 3 使用 `figure` 环境，所以它们可以和 endfloat 宏包（见第 18.5 节）一起使用。

§ 23.1. Landscape 环境

landscape 宏包（包括在标准的 LaTeX 图形宏包套件中）定义了 `landscape` 环境，允许在竖排版的文档中放置横排页。横排页被旋转使得竖排页的左边界为其顶部。

输入命令 `\begin{landscape}` 使得所有未处理的竖排的浮动对象被排出并开始横排页，同样地，输入命令 `\end{landscape}` 使得所有未处理的横排的浮动对象被排出并重新回到竖排状态。

所有位于 `landscape` 环境中的内容都会被横排。如果只有包含一个浮动图形环境

```
\begin{landscape}
  \begin{figure}
    \centering
    \includegraphics[width=4in]{graphic.eps}
    \caption{Landscape Figure}
  \end{figure}
\end{landscape}
```

这时会得到如图 23.1 所示的横排图形。不过，由于 `landscape` 开始一新页，可能会导致页面出现很大空白。如同此页:)

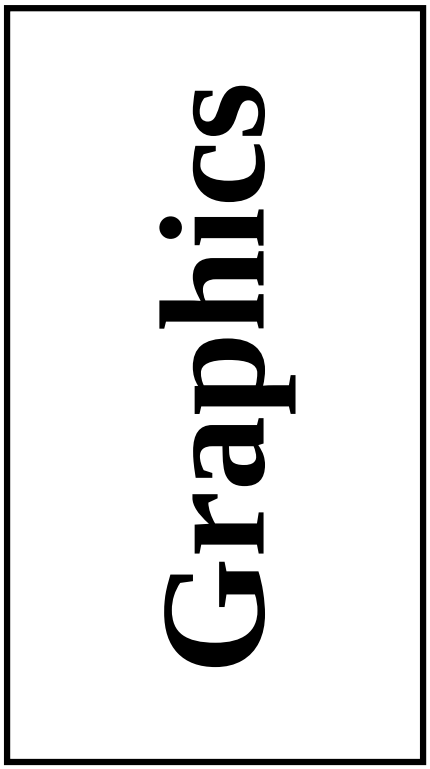


图 **23.1:** Landscape Figure

§ 23.2. Sidewaysfigure 环境

rotating 宏包提供了 sidewaysfigure 环境来生成横排的图形。例如：

```
\begin{sidewaysfigure}
  \centering
  \includegraphics[width=4in]{graphic.eps}
  \caption{Sidewaysfigure Figure}
\end{sidewaysfigure}
```

得到图 23.2。

与 landscape 环境不同的是，由 sidewaysfigure 得到的图形可在竖排页中浮动以避免导致出现过多空白的页面。相反 landscape 环境则有更大的灵活性，允许横排页中有文本，表格和图形等。

sidewaysfigure 排出的图形的外观缺省由文档使用 oneside 还是 twoside 版式所决定。

- 当使用 oneside 时，图形的底部面向竖排页的右边界。
- 当使用 twoside 时，图形的底部面向竖排页的外边界。

在调入 rotating 是使用宏包选项可以改变上述缺省行为。如：

```
\usepackage[figuresleft]{rotating}
```

使得用 sidewaysfigure 排出的图形的底部面向竖排页的左边界（无论是 oneside 还是 twoside ）。同样，

```
\usepackage[figuresright]{rotating}
```

使得用 sidewaysfigure 排出的图形的底部面向竖排页的右边界。

§ 23.3. Rotcaption 命令

用第 23.1 节和第 23.2 节的方法得到的横排图形都是放在一单独的横排页上的。不过对于比较小的图形来说，显然没有必要。这种情况下，可以利用 rotating 宏包中的 \rotcaption 来得到小的横排图形。例如：

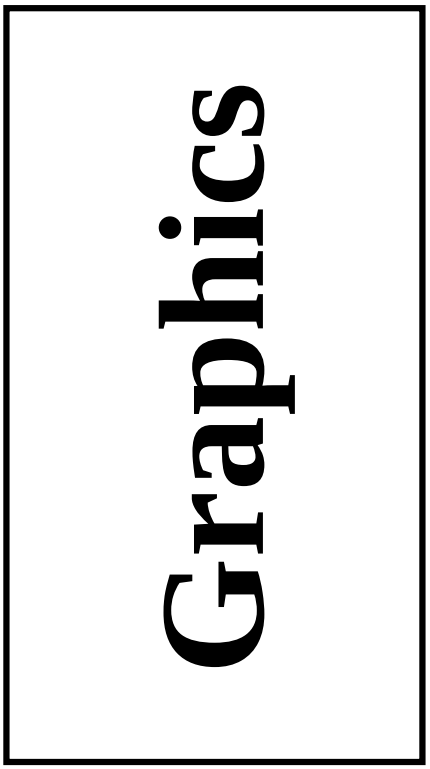


图 23.2: Sidewaysfigure Figure

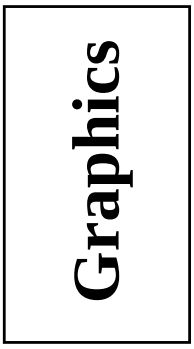


图 23.3: Rotcaption Caption

```
\begin{figure}
  \centering
  \begin{minipage}[c]{1in}
  \includegraphics[angle=90,width=\textwidth]{graphic.eps}
  \end{minipage}
  \begin{minipage}[c]{0.5in}
  \rotcaption{Rotcaption Caption}
  \label{fig:rotcaption}
  \end{minipage}
\end{figure}
```

得到图 23.3。

\rotcaption 命令生成的标题总是旋转使得其底部面向页面的右边界。与第 23.1 节和第 23.2 节的方法不同的是， \rotcaption 并不旋转图形。因此上例中的 \includegraphics 命令需要使用 angle=90 这一选项。

24

标 题 在 一 边 的 图 形

一般来说，图形的标题放置在其上方或下方。本章将介绍怎样将标题放置在图形的旁边[1]。第 24.1 节介绍了将标题置于图形左侧的方法，同样地也可将标题置于图形的右侧。对于双面版式的文档，第 24.2 节介绍了将标题置于图形内侧（奇数页中为图形的左侧，偶数页中为图形的右侧）的方法。

§ 24.1. 图形左侧标题

\caption 命令一般将标题置于图形或表格的下方。可以利用小页环境来欺骗 \caption 命令，从而使它把标题放在图形的一侧。例如命令：

```
\begin{figure}
  \centering
  \begin{minipage}[c]{.45\textwidth}
    \centering
    \caption{Caption on the Side}
    \label{fig:side:caption}
  \end{minipage}%
  \begin{minipage}[c]{.45\textwidth}
    \centering
    \includegraphics[width=\textwidth]{graphic.eps}
```

[1]因为 float 宏包定义的 figure 环境中，标题固定在图形的下方，因此无法使用它来得到置于图形旁边的标题。只要没有声明 \restylefloat 命令，其它的 float 宏包的命令都可使用。

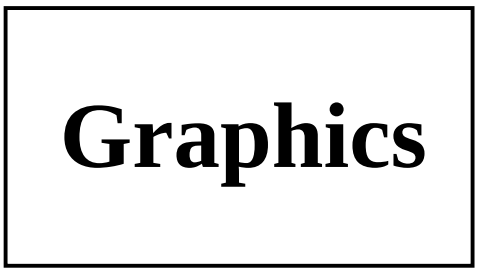


图 24.1: Caption on the Side

```
  \end{minipage}
\end{figure}
```

得到如图 24.1 的结果。在小页之间加入像 \hfill 或 \hspace{.05\textwidth} 的水平距离可能会更好些。

图 24.1 中标题和图形垂直居中。如果想让图形和标题顶部对齐或底部对齐，可参见第 11.4节。

§ 24.2. 图形内侧标题

上节图 24.1 中将标题放在图形的左侧，而对于双面版式的文档，常常会希望将标题置于图形的内侧。这时可用 ifthen 宏包的 \ifthenelse 命令来指定对奇数页和偶数页所使用的不同代码。例如：

```
\usepackage{ifthen}
...
\begin{figure}
  \centering
  \ifthenelse{\isodd{\pageref{fig:side:caption}}}
  {% BEGIN ODD-PAGE FIGURE
  \begin{minipage}[c]{.45\textwidth}
    \centering
    \caption{Caption on the Side}
    \label{fig:side:caption}
  \end{minipage}%
  \hspace{0.05\textwidth}%
  \begin{minipage}[c]{.45\textwidth}
    \includegraphics[width=\textwidth]{graphic.eps}
```

```
  \end{minipage}%
  }% END ODD-PAGE FIGURE
  {% BEGIN EVEN-PAGE FIGURE
  \begin{minipage}[c]{.45\textwidth}
    \includegraphics[width=\textwidth]{graphic.eps}
  \end{minipage}%
  \hspace{0.05\textwidth}%
  \begin{minipage}[c]{.45\textwidth}
    \centering
    \caption{Caption on the Side}
    \label{fig:side:caption}
  \end{minipage}%
  }% END EVEN-PAGE FIGURE
\end{figure}
```

生成的图形其标题总在图形的内侧。

§ 24.3. Sidecap 宏包

利用前面量节介绍的方法可以得到标题在一侧的图形。如果希望有更多的灵活性，那么使用 sidecap 宏包将更为简单方便。

当在 sidecap 宏包提供的 `SCfigure` 环境中使用 `\caption` 命令时，标题会被自动地放置于图形的一侧。例如：

```
\usepackage{sidecap}
...
\begin{SCfigure}
  \includegraphics[width=3in]{graphic.eps}
  \caption{This is a SCfigure}
\end{SCfigure}
```

结果如图 24.2 所示。

sidecap 宏包在用 `\usepackage` 调入时有下面四个可选项：

outercaption 标题在偶数页中出现在左侧，奇数页中出现在右侧。这也是 sidecap 宏包的缺省选项。

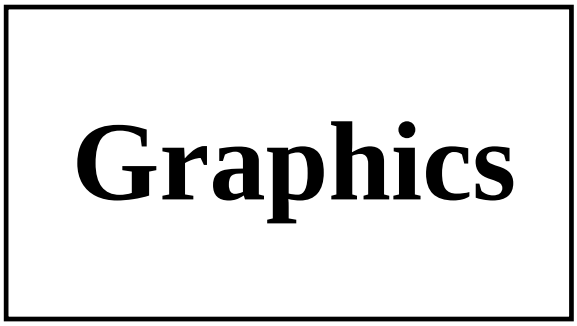


图 24.2: This is a SCfigure

innercaption 标题在偶数页中出现在右侧，奇数页中出现在左侧。

leftcaption 标题总出现在左侧。

rightcaption 标题总出现在右侧。

`Scfigure` 环境包括下面两个可选参数：

- 第一个可选参数指定标题对于图形的相对宽度。一个大的值（如 100）会让标题使用最大可能的宽度。缺省为 1。
- 第二个可选参数指定图形的浮动位置选项。如 `[htp]` 或 `[!ht]` 等，详见第 refsec:figplacement 节。

25

奇 偶 页 中 的 图 形

图形环境的浮动放置算法不能控制图形出现在奇数页还是偶数页。要达到控制浮动图形的奇数或偶数页放置，必须使用 afterpage 宏包的 \afterpage 命令和 ifthen 宏包的 \ifthenelse 命令。

将图形置于 figure 环境，可能会使得在偶数页中声明的图形被浮动到奇数页中。反之，使用第 20 章中定义的 \figcaption 命令则可在不用 figure 环境的情况下生成图形。

```
\makeatletter
  \newcommand\figcaption{\def\@captype{figure}\caption}
\makeatother
```

使用 \ifthenelse 命令可用来将出现在奇数页上的图形放到下一偶数页上。这需要重复一次插图命令，一次是对应于下一页为奇数页的情况，另一次则对应于下一页为偶数页的情况。为简便起见，首先定义一个 \leftfig 命令：

```
\newcommand\leftfig{%
  \vspace*{\fill}%
  \centering
  \includegraphics{graphic.eps}
  \figcaption{This is on the left (even) page.}
  \vspace*{\fill}\newpage}
```

接下来就可以用这个新定义的命令和 \afterpage,\ifthenelse 命令一起来生成一幅只出现在偶数页上的图形。

```
\afterpage{\clearpage%
  \ifthenelse{\isodd{\value{page}}}%
    {\afterpage{\leftfig}}%
    {\leftfig}}
```

几点说明：

- 欲使图形只出现在奇数页上，掉换一下 \ifthenelse 的参数顺序即可。

```
\afterpage{\clearpage%
  \ifthenelse{\isodd{\value{page}}}%
    {\leftfig}}%
    {\afterpage{\leftfig}}
```

- 使用 \value{page} 而不是 \pageref 的好处是它总是正确的。相反，\pageref 只有在 LaTeX 的交叉引用收敛时才正确。
- 当图形较大时，可能会出现在图形中间（图形与标题之间）分页的情况。这时可将它放到一个小页环境中以保持它的完整性。

```
\newcommand\leftfig{%
  \vspace*{\fill}%
  \begin{minipage}{\textwidth}
    \centering
    \includegraphics{graphic.eps}
    \figcaption{This is on the left (even) page.}
  \end{minipage}
  \vspace*{\fill}\newpage}
```

- \afterpage 命令在极少数情况下会造成一个 “lost float” 的错误，这时将 \clearpage 从 \ifthenelse 前去掉可能会有所帮助。

```
\afterpage{\ifthenelse{\isodd{\value{page}}}%
  {\afterpage{\leftfig}}%
  {\leftfig}}
```

- 在上面的例子中，图形是占据完整的一偶数页的。要将其置于偶数页的顶部，修改或去掉 \vspace*{\fill} 和 \newpage 命令。

```
\newcommand\leftfig{%
  \centering
  \includegraphics{graphic.eps}
  \figcaption{This is at the top of the left (even) page.}
  \vspace{\floatsep}}
```

§ 25.1. 迎面页图形

在双面版式的文档中，为消除浮动图形间差别，常常希望将图形放在迎面页（facing page）上。为达到这一目的，仍须使用与前两节中相似的方法。为简单起见，定义命令 \facingfigures 如下：

```
\newcommand\facingfigures{%
  \vspace*{\fill}%
  \centering
  \includegraphics{left.eps}
  \figcaption{This is on the left (even) page.}
  \vspace*{\fill}\newpage\vspace*{\fill}%
  \centering
  \includegraphics{right.eps}
  \figcaption{This is on the right (odd) page.}
  \vspace*{\fill}\newpage}
```

这时可用 \facingfigures 与 \afterpage,\ifthenelse 一起来生成迎面页图形。

```
\afterpage{\clearpage%
  \ifthenelse{\isodd{\value{page}}}%
   {\afterpage{\facingfigures}}%
   {\facingfigures}}
```

26

盒 子 中 的 图 形

盒子中的图形通常指下面两种情形：

- 图形在盒子中，但其标题在盒子之外。
- 图形及其标题都在盒子中。

将某一对象置于盒子中的最基本的方法就是把它放到 \fbox 命令中，这样会将该对象用一长方形的框围起来。 fancybox 宏包提供了不同式样的盒子。

§ 26.1. 图形在盒子中

把 \includegraphics 命令放到 \fbox 中会使所插入的图形置于一个带框盒子中。例如：

```
\begin{figure}
  \centering
  \fbox{\includegraphics[totalheight=2in]{pend.eps}}
  \caption{Box Around Graphic, But Not Around Caption}
  \label{fig:boxed_graphic}
\end{figure}
```

如图 26.1所示，图形被置于一带框盒子中。

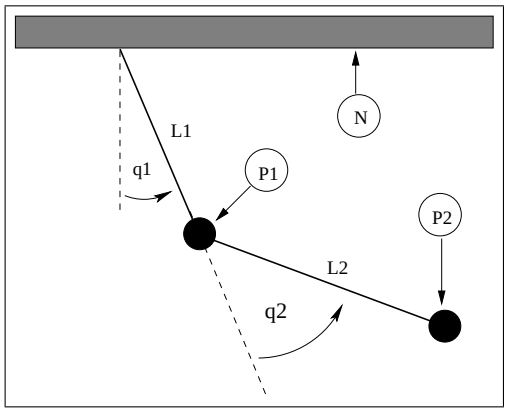


图 26.1: Box Around Graphic, But Not Around Caption

§ 26.2. 图形与标题均在盒子中

要将图形与标题均置于盒子中，也许有人想当然的以为把 \caption 命令也放到 \fbox 命中就可以了。然而，由于 \caption 命令只能在段落模式中使用，而 \fbox 命令中的内容是在 LR 模式中被处理的[1]，所以这样做是不能达到目的的。

因为小页环境的内容和 \parbox 命令都使在段落模式中处理，所以将 \fbox 命令的内容放到小页环境或 \parbox 命令中，就可以把 \caption 包含在 \fbox 中。由于小页环境和 \parbox 命令都必须给出它们的宽度，故没有直接的办法让 \fbox 和图形及其标题一样宽。例如下列命令：

```
\begin{figure}
  \centering
  \fbox{
    \begin{minipage}{4 in}
      \centering
      \includegraphics[totalheight=2in]{pend.eps}
      \caption{Box Around Figure Graphic and Caption}
      \label{fig:boxed_figure}
    \end{minipage} }
\end{figure}
```

[1] LaTeX 使用三种模式， LR ， 段落模式和数学模式。

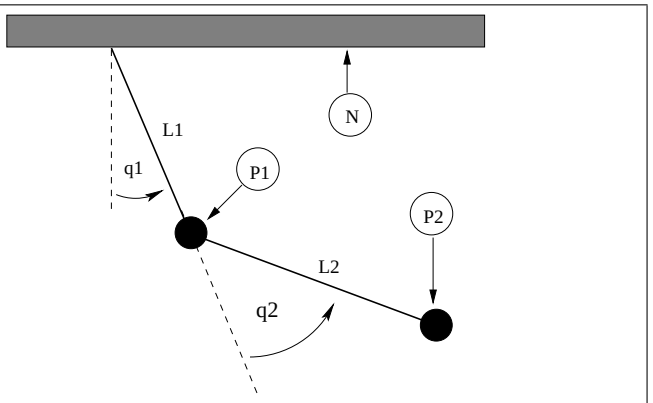


图 26.2: Box Around Figure Graphic and Caption

得到图 26.2，其中图形与标题都置于盒子中。

一般通过不断的尝试修改来确定小页环境的宽度从而使得盒子能够恰好围住图形和标题。下面的这些方法可以避免枯燥麻烦的尝试修改。

1. 选择一个确定的小页的宽度，使得图形的宽度与其相同。

```
\includegraphics[width=\textwidth]{pend.eps}
```

2. 当指定图形的高度时，适当的小页的宽度可以通过把图形放到一个盒子中，然后计算盒子的宽度来得到。

```
\newsavebox{\mybox}
\newlength{\mylength}
\sbox{\mybox}{\includegraphics[height=3in]{file.eps}}
\settowidth{\mylength}{\usebox{\mybox}}
\begin{figure}
  \centering
  \fbox{
    \begin{minipage}{\mylength}
      \centering
      \usebox{\mybox}
      \caption{Box Around Figure Graphic and Caption}
      \label{fig:boxed_figure}
    \end{minipage} }
\end{figure}
```

3. 为保证标题只有一行，可以使用 \settowidth 命令来估计标题的宽度并将其作为小页的宽度。

```
\newlength{\mylength}
\settowidth{\mylength}%
  {Figure XX: Box Around Figure Graphic and Caption}
\fbox{ \begin{minipage}{\mylength}
  ...
```

§ 26.3. 定制 fbox 的参数

在图 26.1 和 26.2 中，盒子由厚为 0.4pt 的直线围成，在框线和图形之间有 3pt 的距离。这些维数值都可以通过使用 \setlength 命令设置 LaTeX 的长度变量 \fboxrule 和 \fboxsep 来修改。例如命令：

```
\begin{figure}
  \centering
  \setlength{\fboxrule}{3pt}
  \setlength{\fboxsep}{1cm}
  \fbox{\includegraphics[totalheight=2in]{pend.eps}}
  \caption{Graphic with Customized Box}
  \label{fig:boxed_custom}
\end{figure}
```

使得盒子的边框线厚为 3pt 且其与图形间的距离为 1 厘米。如图 26.3 所示。

§ 26.4. fancybox 宏包

在图 26.1,26.2 和 26.3 中，用 \fbox 命令将图形包围在标准的长方形框盒子中。要想使用不同类型的盒子，可使用 fancybox 宏包。它提供了 \shadowbox, \doublebox, \ovalbox 和 \Ovalbox 四个命令来生成不同形状的盒子。

如同 \fbox 命令一样，这些盒子命令中的内容与边框间距由 LaTeX 长度 \fboxsep 控制。长度 \shadowsize 可用 \setlength 命令来设定。而 \ovalbox 和 cmdOvalbox 命令中的边框线厚度对应于 picture 环境中的 \thinlines 和

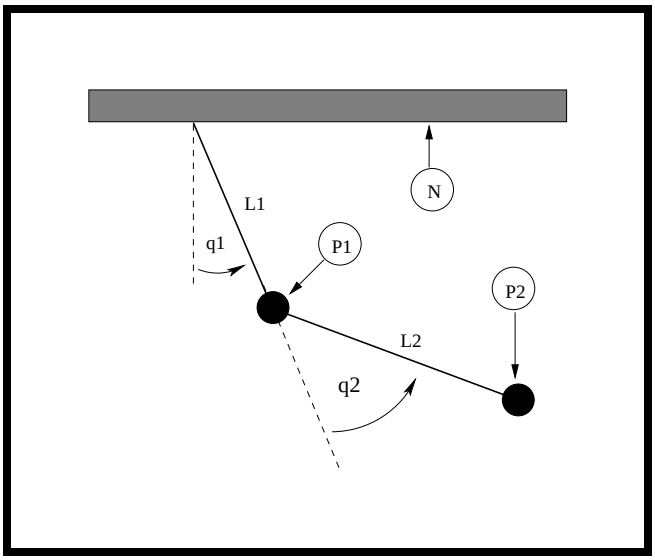


图 26.3: Graphic with Customized Box

表 26.1: FancyBox Commands

\shadowbox{Example} Example	• 盒子边框线厚度为 \fboxrule • 盒子阴影厚度为 \shadowsize（缺省为 4pt）。
\doublebox{Example} Example	• 内框线厚为 .75\fboxrule。 • 外框线厚为 1.5\fboxrule。 • 内外框之间的距离为 1.5\fboxrule+0.5pt。
\ovalbox{Example} Example	• 盒子边框线厚度为 \thinlines。 • 使用 \cornersize{x} 四个角的直径设为 x 乘以盒子宽和高之间较小的那个。缺省为 0.5。 • 使用 \cornersize*{x} 命令直接将四个角的直径设为 x。如 \cornersize*{1cm} 将四个角的直径设为 1 厘米。
\Ovalbox{Example} Example	除了盒子边框线厚度为 \thicklines 外，均与 \ovalbox 一样。

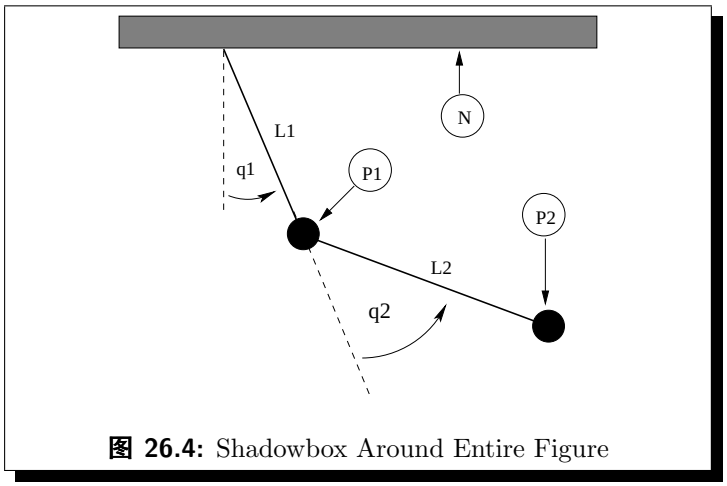


图 26.4: Shadowbox Around Entire Figure

\thicklines 的值，由于它们不是长度，所以无法用 \setlength 来设定。这两个值依赖于当前字体的大小和形状，缺省分别为 0.4pt 和 0.8pt。例如：

```
\begin{figure}
  \centering
  \shadowbox{
    \begin{minipage}{3.5 in}
      \centering
      \includegraphics[totalheight=2in]{pend.eps}
      \caption{Shadowbox Around Entire Figure}
      \label{fig:boxed_fancy}
    \end{minipage} }
\end{figure}
```

用一个带阴影的盒子将图形与标题包围起来，如图 26.4 所示。

27

并 列 的 图 形

使图形并列所需的命令依赖于用户到底想怎样来组织图形。本章主要讨论三种常见的并列图形。

1. 多个图形并列于一个图形环境中。
2. 多个并列的浮动图形，如图 27.3 和 27.4。
3. 一图形环境中各个子图的平行排列。如子图 27.9(a) 和 27.9(b) 并列于图 27.9 中。

本章中将用下列两种方法来生成上述三种并列图形。

1. 连续使用 `\includgraphics` 命令。
2. 并列的小页环境，其中每个都包含一 `\includegraphics` 命令。

理解第 27.2 节的内容在构造多个并列的浮动图形非常重要。并列的浮动图形是通过将盒子（`\incudegraphics` 或小页）平行放置在一条线上来得到的。

§ 27.1. 一图形环境中的并列图形

连续使用多个 `\includgraphics` 命令是生成并列图形的最简单的方法，仅管使用并列的小页环境能够让那些并列的图形更好地对齐。



图 27.1: Two Graphics in One Figure

下面的命令：

```
\begin{figure}
  \centering
  \includegraphics[width=1in]{graphic.eps}%
  \hspace{1in}%
  \includegraphics[width=2in]{graphic.eps}
  \caption{Two Graphics in One Figure}
\end{figure}
```

得到如图 27.1 的并列图形。4 英寸宽，居中放置。其中的 \hspace 命令可用 \hfill 来代替，使得将图形推向页面的两边边界（见第 10.2 节）。

将 \includegraphics 命令放到小页环境中可以让用户更好地控制图形的对齐方式。例如：

```
\begin{figure}
  \centering
  \begin{minipage}[c]{0.5\textwidth}
    \centering
    \includegraphics[width=1in]{graphic.eps}
  \end{minipage}%
  \begin{minipage}[c]{0.5\textwidth}
    \centering
    \includegraphics[width=2in]{graphic.eps}
  \end{minipage}
  \caption{Centers Aligned Vertically}
\end{figure}
```

生成图 27.2，其中的图形是中间对齐的。

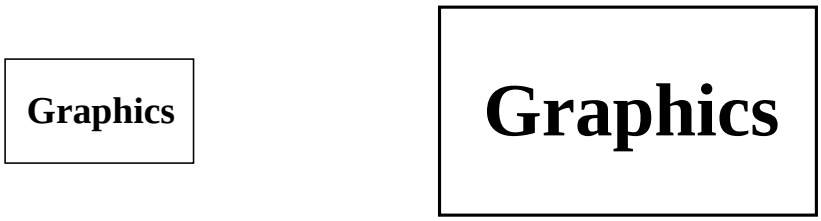


图 27.2: Centers Aligned Vertically

对于这个例子，需要注意以下几点：

- 如同其它的 LaTeX 对象一样，小页在放置时，它的参考点和当前基线对齐。缺省小页使用 [c] 选项，将参考点置于其竖直方向的中点。其它的选项如 [t],[b] 等的含义与使用技巧可参见第 11.4 节。
- 在第一个 \end{minipage} 后面的 % 防止在两个小页盒子中间加上一个字符间距，详见第 10.2 节。
- 当几个并列小页的宽度之和没有达到 1.0\textwidth 时，可用 \hspace 或 \hfill 来确定水平间距，详见第 10.2 节。

§ 27.2. 并列的浮动图形

在上一节中，通过在一个图形环境中使用多个小页环境从而得到一个由多幅图形组成的浮动图形。若将 \caption 命令放到每个小页环境中，则每个小页环境就生成一浮动图形。例如：

```
\begin{figure}
  \begin{minipage}[t]{0.5\linewidth}
    \centering
    \includegraphics[width=1in]{graphic.eps}
    \caption{Small Box}
    \label{fig:side:a}
  \end{minipage}%
  \begin{minipage}[t]{0.5\linewidth}
    \centering
    \includegraphics[width=1.5in]{graphic.eps}
```



图 27.3: Small Box

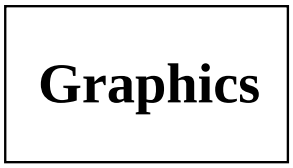


图 27.4: Big Box

```
    \caption{Big Box}
    \label{fig:side:b}
  \end{minipage}
\end{figure}
```

生成图 27.3 和 27.4。尽管上面的命令只使用了一个 `figure` 环境，但由于每个小页中都包含一个 `\caption` 命令，所以仍然得到两个浮动图形。

在图 27.3 和 27.4中，并列的小页环境使用了 `[t]` 选项，使得两幅图形的基线对齐。这对于非旋转的图形没有任何问题，而且使得两标题的顶部对齐。不过，如果图形的底部不对齐的话（如其中一图形被旋转），就会发生问题。例如：

```
\begin{figure}
  \centering
  \begin{minipage}[t]{.33\textwidth}
    \centering
    \includegraphics[width=2cm]{graphic.eps}
    \caption{Box with a Long Caption}
  \end{minipage}%
  \begin{minipage}[t]{.33\textwidth}
    \centering
    \includegraphics[width=2cm,angle=-30]{graphic.eps}
    \caption{Rotated Box}
  \end{minipage}%
\end{figure}
```

生成图 27.5 和 27.6，我们可以看到这里两幅图形的标题并不对齐。而若只使用小页的 `[b]` 选项，会使得标题的最后一行对齐，并不能解决问题。

一种解决办法是在小页环境中把图形和标题分开放到两行中：第一行放置图形，第二行放置标题。例如：



图 27.5: Box with a Long Caption

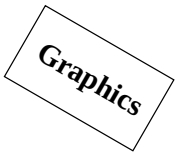


图 27.6: Rotated Box



图 27.7: Box with a Long Caption

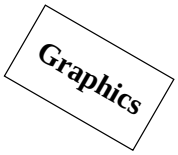


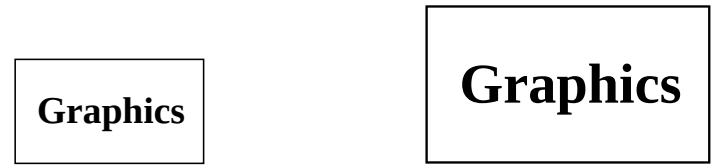
图 27.8: Rotated Box

```
\begin{figure}
  \centering
  \begin{minipage}[b]{.33\textwidth}
    \centering
    \includegraphics[width=2cm]{graphic.eps}
  \end{minipage}%
  \begin{minipage}[b]{.33\textwidth}
    \centering
    \includegraphics[width=2cm,angle=-30]{graphic.eps}
  \end{minipage}\\[-10pt]
  \begin{minipage}[t]{.33\textwidth}
    \caption{Box with a Long Caption}
  \end{minipage}%
  \begin{minipage}[t]{.33\textwidth}
    \caption{Rotated Box}
  \end{minipage}%
\end{figure}
```

生成的图 27.2 和 27.8 中，图形的基线和标题的第一行分别对齐。

在这个例子中，需要注意：

- 在最后一幅图后面用 \\ 来断行， \\ 的参数项 [-10pt] 使得图形与标题之间的距离比当前行距减少 10pt。这样做是让图形和标题更接近些，用户也可自己选用合适的值。



(a) Small Box with a Long Caption

(b) Big Box

图 27.9: Two Subfigures

- 包含图形的小页使用 [b] 选项，使得它们的参考点为其最后一行的基线。
- 包含标题小页使用 [t] 选项，使得它们的参考点为其第一行的基线。
- 任何一个 \label 命令都必须和它相应的 \caption 命令在同一个小页中。

§ 27.3. 并列的子图形

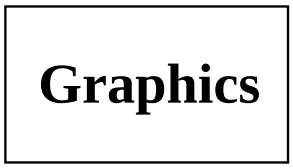
在某些情况下，有时会希望将并列的图形组成一组，而其中的每一幅图都保持其独立性。 paisubfigure 宏包的 \subfigure 命令将这一组做为一幅图形，其中的每一幅图做为子图形。例如：

```
\begin{figure}
  \centering
  \subfigure[Small Box with a Long Caption]{
    \label{fig:subfig:a} %% label for first subfigure
    \includegraphics[width=1.0in]{graphic.eps}}
  \hspace{1in}
  \subfigure[Big Box]{
    \label{fig:subfig:b} %% label for second subfigure
    \includegraphics[width=1.5in]{graphic.eps}}
  \caption{Two Subfigures}
  \label{fig:subfig} %% label for entire figure
\end{figure}
```

生成图 27.9。这里使用 LaTeX 的引用命令 \ref{fig:subfig:a} 会得到 27.9(a)，\ref{fig:subfig:b} 得到 27.9(b)， \ref{fig:subfig} 得到 27.9。



(a) Small Box with a Long Caption



(b) Big Box

图 27.10: Minipages Inside Subfigures

像其它的并列图形一样，子图也可以在小页环境中使用。而且在一些情况下，这样做还能更方便的得到理想的图形间距。例如：

```
\begin{figure}
  \subfigure[Small Box with a Long Caption]{
    \label{fig:mini:subfig:a} %% label for first subfigure
    \begin{minipage}[b]{0.5\textwidth}
      \centering
      \includegraphics[width=1in]{graphic.eps}
    \end{minipage}}%
  \subfigure[Big Box]{
    \label{fig:mini:subfig:b} %% label for second subfigure
    \begin{minipage}[b]{0.5\textwidth}
      \centering
      \includegraphics[width=1.5in]{graphic.eps}
    \end{minipage}}
  \caption{Minipages Inside Subfigures}
  \label{fig:mini:subfig} %% label for entire figure
\end{figure}
```

得到图 27.10，其中包括两个子图 27.10(a) 和 27.10(b)。

图 27.10 中的子图标题比图 27.9 中的要宽一些。这是因为子图标题的宽度和子图的宽度相同，图 27.9 中的子图只包含图形，而图 27.10 中的子图包含了宽度为 `0.5\textwidth` 的小页。

子图的标记有两种形式：

1. 一种是出现在子图的下面作为标题的一部分。这通过命令 `\@thesubfigure` 来生成。

2. 另一种是在使用 \ref 命令的时候出现。这通过将命令 \p@subfigure 的输出处理后传递给 \thesubfigure 命令来生成。

上面的这些命令使用 subfigure 计数器和 \thefigure 命令。子图的标记的格式由下面的命令来控制。

- 命令 \thefigure 印出当前图形的编号。
- 计数器 subfigure 记录子图的编号，命令 \alph{subfigure} 将计数器 subfigure 的值用小写字母印出，而命令 \roman{subfigure} 则是用小写罗马数字印出（有关印出计数器值的命令可参见文献 [1, 第 98 页] 和 [3, 第 446 页]。）。
- 命令 \thesubfigure 缺省使用小写字母，如 (a),(b) 等。
- 命令 \@thesubfigure 缺省为 \thesubfigure\space，即在标题标记和文本之间加上一个空白。
- 命令 \p@subfigure 缺省为 \thefigure。

如果改变子图标题的标记，字体等的缺省值，可参见文献 [10]。下面给出几个简单的例子：

子图的例子

- 若想让子图标题标记使用小写罗马数字如 (i), (ii)等， \ref 命令的结果如 12i, 12ii 等，可使用下面的命令（最好放在导言区中）

```
\renewcommand{\thesubfigure}{\roman{subfigure}}
\makeatletter
  \renewcommand{\@thesubfigure}{(\thesubfigure)\space}
  \renewcommand{\p@subfigure}{\thefigure}
\makeatother
```

- 若想让子图标题标记使用阿拉伯数字如 12.1:, 12.2: 等， \ref 命令的结果如 12.1, 12.2 等，可使用下面的命令

```
\renewcommand{\thesubfigure}%
             {\thefigure.\arabic{subfigure}}
```

```
\makeatletter
  \renewcommand{\@thesubfigure}{\thesubfigure:\space}
  \renewcommand{\p@subfigure}{}
\makeatother
```

缺省情况下，用 \listoffigures 命令生成的图形目录中只包括图形，而不包括子图。要想在图形目录中包括子图，要在 \listoffigures 命令前加上 \setcounter{lofdepth}{2}。

需要说明的是，由于 LaTeX 的变化，导致目前版本（*3/95*）的 subfigure 宏包在图形目录的子图输入项开始部分都加上“numberline1”。将下面的代码加到导言区中就可以解决这一问题。

```
\makeatletter
  \renewcommand{\@subcaption}[2]{%
    \begingroup
      \let\label\@gobble
      \def\protect{\string\string\string}%
      \xdef\@subfigcaptionlist{%
        \@subfigcaptionlist,%
        {\numberline {\@currentlabel}%
      \noexpand{\ignorespaces #2}}}%
    \endgroup
  \@nameuse{@make#1caption}{\@nameuse{@the#1}}{#2}}
\makeatother
```

28

堆 叠 图 形

在第 27 章中，通过将几个盒子并排放置在一行中来得到并列图形。堆叠图形（stacked graphics）也可用同样的方法来生成。例如：

```
\begin{figure}
  \centering
  \begin{minipage}[b]{0.3\textwidth}
    \centering
    \includegraphics[width=1in]{graphic.eps}
    \caption{Caption 1}
  \end{minipage}%
  \hspace{0.04\textwidth}%
  \begin{minipage}[b]{0.3\textwidth}
    \centering
    \includegraphics[width=1in]{graphic.eps}
    \caption{Caption 2}
  \end{minipage}\\[20pt]
  \begin{minipage}[b]{0.3\textwidth}
    \centering
    \includegraphics[width=1in]{graphic.eps}
    \caption{Caption 3}
  \end{minipage}%
  \hspace{0.04\linewidth}%
  \begin{minipage}[b]{0.3\textwidth}
    \centering
```



图 28.1: Caption 1



图 28.2: Caption 2



图 28.3: Caption 3



图 28.4: Caption 4



图 28.5: Caption 5

```
    \includegraphics[width=1in]{graphic.eps}
    \caption{Caption 4}
  \end{minipage}%
  \hspace{0.04\linewidth}%
  \begin{minipage}[b]{0.3\textwidth}
    \centering
    \includegraphics[width=1in]{graphic.eps}
    \caption{Caption 5}
  \end{minipage}
\end{figure}
```

得到图 28.1-28.5。其中在 “Caption2” 小页后的 `\\[20pt]` 命令得到一增加了 20pt 的行距。

29

图 形 与 表 格 的 平 行 排 列

在第 27 章中，通过在一个 figure 环境中使用多个 \caption 命令来得到并列的多个图形。同样地，在一个 table 环境中使用多个 \caption 命令可将多个表格平行排列。若想使表格和图形平行排列在一起，可使用第 20 中定义的命令 \figcaption 和 \tabcaption。例如下面的命令：

```
\begin{figure}[htb]
  \begin{minipage}[b]{0.5\textwidth}
    \centering
    \includegraphics[width=0.8\textwidth]{graphic.eps}
    \caption{This is a Figure by a Table}
    \label{fig:by:table}
  \end{minipage}%
  \begin{minipage}[b]{0.5\textwidth}
    \centering
    \begin{tabular}{|c|c|} \hline
      Day & Data \\ \hline\hline
      Monday    & 14.6 \\
      Tuesday   & 14.3 \\
      Wednesday & 14.2 \\
      Thursday  & 14.5 \\
      Friday    & 14.9 \\ \hline
    \end{tabular}
    \tabcaption{This is a Table by a Figure}
```

```
    \label{table:by:fig}
  \end{minipage}
\end{figure}
```

用一个 figure 环境生成了并排放置的图 29.1 和表 29.1。

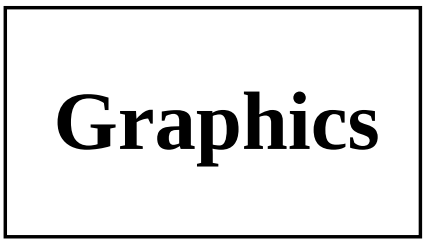


图 29.1: This is a Figure by a Table

Day	Data
Monday	14.6
Tuesday	14.3
Wednesday	14.2
Thursday	14.5
Friday	14.9

表 29.1: This is a Table by a Figure

因为 LaTeX 允许图形的浮动不必考虑其前后表格的顺序，所以在 figure 环境中使用命令 \tabcaption 可能会将表格放置到尚未处理的浮动图形前面。同理，在 table 环境中使用命令 \figcaption 可能会将图形放置到尚未处理的浮动图形前面。这种情况下，可以在图形环境前使用 \FloatBarrier 命令来清除其前面尚未处理的浮动图形。

30

图 文 混 排

在使用外部图形时，通常的是将其置于一个 figure 环境中，由这一浮动环境来决定最后的位置是在页面的上方或下方。但有的时候，许多使用者往往希望将图形放置在一个正文方格内，或者置于页面的左右，也可能是在页面的的中间，四周包围者文本，甚至放在文字的下方作为背景，或重叠放置。这时，前面所介绍的只使用 LaTeX 图形宏包套件就很难得到所希望的结果。本章将介绍几个有用的图形宏包，可以让你很容易地得到上述特殊效果。

本章介绍的几个图形宏包均可从 CTAN 下载。如果你使用的是 teTeX 或 fpTeX，那么这些宏包已包括在内了，你所做的只需是在文档中调用它们：

```
\usepackage[选项]{宏包}
```

除了本章所介绍的宏包外，还有一些宏包也可完成同样的工作。如 floatflt 也可用来将图形置于文本段落的一边。而所介绍的宏包中，也有未涉及的内容，进一步的研究可阅读这些宏包所附的帮助文件。

§ 30.1. Wrapfig 宏包

Wrapfig 宏包提供了一个 wrapfigure 环境[1]来排版窄小的图形，使得该图形位于文本的一边，并使文本在其边上折行。

[1] wrapfig 也同时提供了一个 wraptable 环境。

wrapfigure的用法：

```
\begin{wrapfigure}{行数}[位置][超出长度]{宽度}<图形>\end{wrapfigure }
```

这里 行数是指图形高度所占的文本行的数目。如果不给出此选项， wrapfig 会自动计算。 位置是指图形相对于文本的位置，须给定下面四项的一个。

[r],[R] 表示图形位于文本的左边。

[l],[L] 表示图形位于文本的右边。

[i],[R] 表示图形位于页面靠里的一边（用在双面格式里）。

[o],[O] 表示图形位于页面靠外的一边。

超出长度是指图形超出文本边界的长度，缺省为 0pt。 宽度则指图形的宽度。wrapfig 会自动计算图形的高度。不过，我们也可设定图形的高度，具体可见 wrapfig.sty 内的说明。

在使用 wrapfig 时需要注意下面几点：



- 在 wrapfigure 后必须紧接着输入段落文字，否则会出错。
- 不能在任何列表环境中使用 wrapfigure，也不能在列表环境前后使用，除非两者之间有一空行或分段指令 \par。
- 如果将 wrapfigure 放在 \parbox 或小页环境等分组中，文本折行必须在这些分组前结束。
- 在双栏页版式中不能使用 wrapfigure。
- 如果在 wrapfigure 中使用 figure 等浮动对象，它的编号有可能不正确。
- 如果在 wrapfigure 中使用 table 等浮动对象，它上下方的横线可能被忽略，必须自己再加入。

- 在折行的文本中， \linewidth 并没有改变。

wrapfig 还可用来放大段落的第一个字。本节的第一个字目 W 就是使用如下命令来得到的：

```
\newcommand{\PartSize}{\fontsize{1.5cm}{1.5cm}\selectfont}
\intextsep=0pt
\begin{wrapfigure}{l}{25pt}
\textcolor{blue}{\mbox{\texttt{\PartSize W}}}
\end{wrapfigure}
\noindent\texttt{rapfig}宏包提供了一个...
```

本节中的另一例子使用了如下命令：

```
\begin{wrapfigure}{r}{4.5cm}
\includegraphics [width=4cm,clip]{tiger.ps}
\end{wrapfigure}
\mbox{}在使用\textsf{wrapfig}时需要注意下面几点：
```

§ 30.2. Picinpar 宏包

picinpar 宏包定义了一个基本的环境 window，还有两个变体 figwindow 和 tabwindow。允许在文本段落中打开一个“窗口”，在其中放入图形、文字和表格等。这里我们主要讨论将图形放入文本段落的用法，其它的用法可参考 picinpar 的说明。

```
\begin{window}[行数，对齐方式，内容，内容说明]\end{window}
```

```
\begin{figwindow}[行数，对齐方式，图形，标题]\end{figwindow}
```

这里的 *行数*是指“窗口”开始前的行数。 *对齐方式*是指在段落中“窗口”的对齐方式。缺省为 l，即左对齐。另外两种是 c：居中和 r：右对齐。第三个参数是出现在“窗口”中的内容，这在 figwindow 中就是要插入的图形。第四个参数则是对“窗口”内容的说明性文字，这在 figwindow 中就是图形的标题。下面是几个例子：

```
\begin{window}[2,c,{\fcolorbox{morelight}{\shortstack{%
   \color{yellow} 你在他乡 \\还 好 \\ 吗? }}},{}]
可是哈卜拉姆再聪明……
……可是我偏不喜欢。」
\end{window}
```

可是哈卜拉姆再聪明、再有学问，有一件事却是他不能解答的，因为包罗万有的「可兰经」上也没有答案；如果你深深爱著的人，却深深的爱上了别人，有甚麼法子？白马带著她一步步的回到中原。白马已经老了，只能慢慢的走，但终是能回到中原的。江南有杨柳、桃花，有燕子、金鱼……汉人中有的是英俊勇武的少年，倜傥潇洒的少年……但这个美丽的姑娘就像古高昌国人那样固执：「那都是很好很好的，可是我偏不喜欢。」

你在他乡
还好
吗?

```
\begin{figwindow}[1,r,{\mbox{%
    \includegraphics[width=4cm]{tiger.ps}}},{Tiger}]
可是哈卜拉姆再聪明……
……可是我偏不喜欢。」
\end{window}
```

可是哈卜拉姆再聪明、再有学问，有一件事却是他不能解答的，因为包罗万有的「可兰经」上也没有答案；如果你深深爱著的人，却深深的爱上了别人，有甚麼法子？白马带著她一步步的回到中原。白马已经老了，只能慢慢的走，但终是能回到中原的。江南有杨柳、桃花，有燕子、金鱼……汉人中有的是英俊勇武的少年，倜傥潇洒的少年……但这个美丽的姑娘就像古高昌国人那样固执：「那都是很好很好的，可是我偏不喜欢。」



图 30.1: Tiger

```
\begin{figwindow}[1,c,{\mbox{%
    \includegraphics[width=3cm]{tiger.ps}}},{Tiger}]
可是哈卜拉姆再聪明……
……可是我偏不喜欢。」
\end{window}
```

可是哈卜拉姆再聪明、再有学问，有一件事却是他不能解答的，因为包罗万有的「可兰经」上也没有答案；如果你深深爱著的人，却深深的爱上了别人，有甚麼法子？白马带著她一步步的回到中原。白马已经老了，只能慢慢的走，但终是能回到中原的。江南有杨柳、桃花，有燕子、金鱼……汉人中有的是英俊勇武的少年，倜傥潇洒的少年……但这个美丽的姑娘就像古高昌国人那样固执：「那都是很好很好的，可是我偏不喜欢。」



图 30.2: Tiger

在使用 picinpar 时要注意以下几点：

- 不要在 window 环境中使用 \samepage。
- 不要在 window 环境中使用 \footnote，代之在用 \footnotemark 标记角注，而将角注的内容在 window 环境外用 \footnotetext 来加入。
- 当使用 paiepic 宏包时，要确保在调入 epic 之前将它调入。

§ 30.3. Picins 宏包

picins 宏包定义了一个命令 \parpic命令，允许将图形等 LaTeX 对象放置在文本段落中。并且，设定适当的参数，可把该对象置于一带框的盒子，有阴影的盒子等等。 \parpic 的用法如下：

```
\parpic(宽度，高度)(水平偏移，垂直偏移)[选项][位置]{图形}
```

上面除了图形必须给出外，其余的均可省略。如果宽度和高度均未给出，那么图形将以它的自然大小来嵌入。选项则可取以下的值：

位置项 只能为下面两个中的一个。

l 将图形置于文本段落的左方（这也是缺省值）。

r 将图形置于文本段落的右方。

外观项 只能为下面五个中的一个，可与上述位置项配合使用。

f 将图形置于一个实框盒子中。

d 将图形置于一个虚框盒子中。

o 将图形置于一个圆角框盒子中。

s 将图形置于一个具有阴影效果的盒子中。

x 将图形置于一个具有立体效果的盒子中。

位置仅当给定的宽度和高度与图形的实际大小相差很大的情况下才起作用。若水平或垂直偏移已给出，那么此项也不起作用。缺省位置是将图形置于盒子的中央。也可取以下的值：

l 将图形置于盒子的左方。

r 将图形置于盒子的右方。

t 将图形置于盒子的上方。

b 将图形置于盒子的下方。

另外， picins 宏包还提供了一些命令来控制图形与文本的间距，图形外框的线宽等。详见 picins 宏包所附的说明。下面是几个例子。



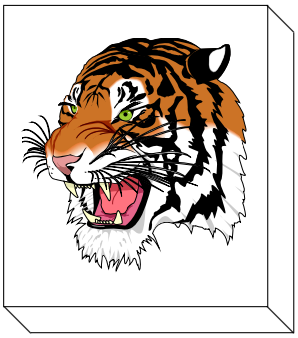
仅当给定的宽度和高度与图形的实际大小相差很大的情况下才起作用。若水平或垂直偏移已给出，那么此项也不起作用。缺省位置是将图形置于盒子的中央。

```
\parpic{%
  \includegraphics[width=3cm]%
                  {tiger.ps}}
仅当给定的宽度和高度与...
```



仅当给定的宽度和高度与图形的实际大小相差很大的情况下才起作用。若水平或垂直偏移已给出，那么此项也不起作用。缺省位置是将图形置于盒子的中央。

```
\parpic(3cm,3.5cm)[sr]{%
  \includegraphics[width=2.5cm]%
                  {tiger.ps}}
仅当给定的宽度和高度与...
```



仅当给定的宽度和高度与图形的实际大小相差很大的情况下才起作用。若水平或垂直偏移已给出，那么此项也不起作用。缺省位置是将图形置于盒子的中央。

```
\boxlength{10pt}%
\parpic(3.5cm,4cm)[xr]{%
  \includegraphics[width=3cm]%
                  {tiger.ps}}
仅当给定的宽度和高度与...
```

31

连 续 图 形

当两个相邻的图形含有关系较为密切的材料时，常常希望具有相同的图形编号。因为计数器 figure 中记录了下一图形的编号，所以可在图形环境前减低 figure 的值使得两幅图形具有相同的编号。例如：

```
\addtocounter{figure}{-1}
\begin{figure}
```

不过，这样做会使得两幅图形无法被正确区分，导致 LaTeX 的引用等的混乱。

构造连续图形的最好的方法是使用 subfigure 宏包。这样既可以使连续的几幅图形具有相同的编号，如 “**图** 12”，且其中的每幅图形也有自己的标记，如 “**图** 12(a)” 等。由于连续的子图位于不同的 figure 环境，所以在两个图形环境之间，必须减低计数器 figure 的值。

```
\addtocounter{figure}{-1}
```

同时，必须在第二幅子图前将子图的计数器 subfigure 加一。

```
\addtocounter{subfigure}{1}
```

例如下面的命令得到两个连续的子图。

```
\begin{figure}
  \centering
```



(a) First Part

图 31.1: Large Graphics



(b) Second Part

图 31.1: Large Graphics (con't)

```
  \subfigure[First Part]{%
    \label{fig:graphics:a}% label for subfigure
    \includegraphics[width=\textwidth]{wide.eps}}%
  \caption{Large Graphics}%
  \label{fig:graphics}% label for figure
\end{figure}
\addtocounter{figure}{-1}
\begin{figure}
  \addtocounter{subfigure}{1}
  \centering
  \subfigure[Second Part]{%
    \label{fig:graphics:b}% label for subfigure
    \includegraphics[width=\textwidth]{wide.eps}}%
  \caption{Large Graphics (con't)}%
\end{figure}
```

在这一例子中，每个图形环境中只有一个子图。而当像第 27.3 节中那样每个图形环境中有多个子图，就需要根据第一个图形环境中子图的个数来相应地调整计数器 subfigure 的增加值。另外，由于连续图形都是不同的浮动对像，有可能不出现在连续的页面上。如果出现这种情况，可在最后一幅连续图形后使用命令 \FloatBarrier 来迫使 LaTeX 将连续图形放置在一起。

参 考 文 献

[1] Leslie Lamport, *LaTeX: A Document Preparation System*, Addison-Wesley, Reading, Massachusetts, second edition, 1994, ISBN 0-201-52983-1

[2] Helmut Kopka and Patrick Daly, *A Guide to LaTeX 2ε*, Addison-Wesley, Reading, Massachusetts, 1995, ISBN 0-201-42777-X

[3] Michel Goossens, Frank Mittelbach and Alexander Samarin, *The LaTeX Companion*, Addison-Wesley, Reading, Massachusetts, 1994, ISBN 0-201-54199-8

[4] Michel Goossens, Sebastian Rahtz and Frank Mittelbach, *The LaTeX Graphics Companion*, Addison-Wesley, Reading, Massachusetts, 1997, ISBN 0-201-85469-4

[5] D. P. Carlisle, *Packages in the 'graphics' bundle* (Documents the `graphics, graphicx, lscape, color` packages), Available as `CTAN/macros/latex/packages/graphics/grfguide.ps`

[6] Tobias Oetiker, *The Not So Short Introduction to LaTeX 2ε*, Available as `CTAN/info/lshort/lshort2e.pdf` and `CTAN/info/lshort/lshort2e.600.ps`

[7] Michel C. Grant and David Carlisle, *The PSfrag system, version 3*, Available as `CTAN/macros/latex/contrib/supported/psfrag/pfgguide.ps`

[8] David Carlisle, *The* ifthen *package*, Available as `CTAN/macros/latex/base/ifthen.dtx`

[9] David Carlisle, *The* afterpage *package*, Available as
CTAN/macros/latex/packages/tools/afterpage.dtx

[10] Steven Douglas Cochran, *The* subfigure *package*, Available as
CTAN/macros/latex/contrib/supported/subfigure/subfigure.dtx

[11] Harald Axel Sommerfeldt, *The* caption *package*, Available as
CTAN/macros/latex/contrib/supported/caption/caption2.dtx

[12] Piet van Oostrum, *Page layout in LaTeX*, Available as
CTAN/macros/latex/contrib/supported/fancyhdr/fancyhdr.tex

[13] Leonor Barroca, *The* rotating *package*, Available as
CTAN/macros/latex/contrib/supported/rotating/rotating.dtx

[14] Timothy Van Zandt, *Documentation for fancybox.sty*, Available as
CTAN/graphics/pstricks/origdoc/fancybox.doc

[15] Donald Arseneau, *The* placeins *package*, Available as
CTAN/macros/latex/contrib/other/misc/placeins.sty

[16] *The* flafter *package*, Available as
CTAN/macros/latex/unpacked/flafter.sty

[17] Don Hosek, *The* morefloats *package*, Available as
CTAN/macros/latex209/contrib/misc/morefloats.sty

[18] James Darrell McCauley and Jeff Goldberg, *The* endfloat *package*, Available as CTAN/macros/latex/contrib/supported/endfloat/endfloat.dtx

索 引

\abovecaptionskip, 104
\AddToShipoutPicture, 84
afterpage, 94
\afterpage, 94

baseline, 5
\belowcaptionskip, 104
\bottomfigurerule, 102
buffersize, 10

calc, 27
caption, 109
\caption, 90
caption2, 105
\captionfont, 116
\captionindent, 110
\captionlabel, 117
\captionlabeldelim, 115
\captionlabelfont, 116
\captionstyle, 110
\centering, 41
\centerline, 42
\clearpage, 93
color, 70
\colorbox, 69

commands
\abovecaptionskip, 104
\AddToShipoutPicture, 84
\afterpage, 94
\belowcaptionskip, 104
\bottomfigurerule, 102
\caption, 90
\captionfont, 116
\captionindent, 110
\captionlabel, 117
\captionlabeldelim, 115
\captionlabelfont, 116
\captionstyle, 110
\centering, 41
\centerline, 42
\clearpage, 93
\colorbox, 69
\cornersize{x}, 151
\DeclareGraphicsExtensions, 35
\DeclareGraphicsRule, 35, 37
\doublebox, 150
\efloatseparator, 107
\epsfbox, 3
\facingfigures, 145

\fancyfoot, 81
\fancyhead, 81
\fancypagestyle, 83
\fbox, 147
\fboxrule, 70
\fboxsep, 70
\fcolorbox, 69
\figurename, 105
\figureplace, 106
\FloatBarrier, 90
\floatpagefraction, 98
\floatsep, 101
\footnotemark, 171
\footnotetext, 171
\graphicspath, 53, 55
\hfill, 42
\HR, 28
\hspace, 42
\hypergetpageref, 131
\ifthenelse, 130
\includegraphics, 3, 4
\kpsewhich, 61
\label, 90
\leavevmode, 42
\leftfig, 143
\listoffigures, 91
\makeatletter, 102
\makeatother, 102
\marginpar, 127
\marginparsep, 127
\marginparwidth, 127
\onelinecaptionfalse, 112
\onelinecaptiontrue, 112
\Ovalbox, 150
\ovalbox, 150
\pageref, 90
\par, 168
\parpic, 172
\psfig, 3
\psfrag, 67
\ref, 90
\reseizebox∗, 33
\resizebox, 32
\restylefloat, 125
\reversemarginpar, 127
\rotatebox, 4, 33
\rotcaption, 133
\scalebox, 4, 32
\setcounter, 97
\settowidth, 150
\shadowbox, 150
\shortstack, 69
\sidewaysfigure, 133
\special, 3
\subfigure, 158
\suppressfloats, 100
\tableplace, 106
\tex, 67, 71
\topfigurerule, 102
\vfill, 101
, 63
\cornersize{x}, 151

current baseline, 5

\DeclareGraphicsExtensions, 35
\DeclareGraphicsRule, 35, 37
depth, 5
\doublebox, 150
doublespace, 121

eco-pic, 84
\efloatseparator, 107
endfloat, 106
environments
 SCfigure, 141
 figure, 90
 figwindow, 169
 landscape, 133
 overpic, 74
 picture, 74
 tabwindow, 169
 window, 169
 wrapfigure, 167
 wraptable, 167
EPS BoundingBox, 8
epsf, 3
\epsfbox, 3
epsfig, 3

\facingfigures, 145
fancybox, 147, 150
\fancyfoot, 81
fancyhdr, 81
\fancyhead, 81
fancyheadings, 81
\fancypagestyle, 83
\fbox, 147
\fboxrule, 70
\fboxsep, 70
\fcolorbox, 69
figure, 90
\figurename, 105
\figureplace, 106
figwindow, 169
flafter, 90, 100
float, 124
\FloatBarrier, 90
floatflt, 167
\floatpagefraction, 98
\floatsep, 101
\footnotemark, 171
\footnotetext, 171

graphics, 3, 25
graphics bundle, 3
\graphicspath, 53, 55
graphicx, 3, 25

header, 8
height, 5
\hfill, 42
\HR, 28
\hspace, 42
\hypergetpageref, 131
hyperref, 131

ifthen, 130
\ifthenelse, 130
\includegraphics, 3, 4

, 61
\kpsewhich, 61

\label, 90
landscape, 133
\leavevmode, 42
\leftfig, 143
\listoffigures, 91
longtable, 133
lscape, 133

\makeatletter, 102
\makeatother, 102
\marginpar, 127
\marginparsep, 127
\marginparwidth, 127
morefloats, 95

\onelinecaptionfalse, 112
\onelinecaptiontrue, 112
\Ovalbox, 150
\ovalbox, 150
overpic, 74
overpic, 74

packages
afterpage, 94
calc, 27
caption, 109
caption2, 105
color, 70
doublespace, 121
eco-pic, 84
endfloat, 106
epsf, 3
epsfig, 3
fancybox, 147, 150
fancyhdr, 81
fancyheadings, 81
flafter, 90, 100
float, 124
floatflt, 167
graphics, 3, 25
graphicx, 3, 25
hyperref, 131
ifthen, 130
longtable, 133
lscape, 133
morefloats, 95
overpic, 74
picinpar, 169
picins, 172
placeins, 90
psfig, 3
PSfrag, 67
rotating, 133
setspace, 121
sidecap, 141
wrapfig, 167, 168
\pageref, 90

\par, 168
\parpic, 172
picinpar, 169
picins, 172
picture, 74
placeins, 90
, 55
psfig, 3
\psfig, 3
PSfrag, 67
\psfrag, 67

\ref, 90
Reference point, 5
\reseizebox∗, 33
\resizebox, 32
\restylefloat, 125
\reversemarginpar, 127
\rotatebox, 4, 33
rotating, 133
\rotcaption, 133

\scalebox, 4, 32
SCfigure, 141
\setcounter, 97
setspace, 121
\settowidth, 150
\shadowbox, 150
\shortstack, 69
sidecap, 141
\sidewaysfigure, 133
\special, 3
\subfigure, 158
\suppressfloats, 100

\tableplace, 106
tabwindow, 169
\tex, 67, 71
\topfigurerule, 102
totalheight, 6
, 93

\vfill, 101

width, 5
window, 169
wrapfig, 167, 168
wrapfigure, 167
wraptable, 167

参考点, 5

当前基线, 5

高度, 5

基线, 5

宽度, 5

全部高度, 6

深度, 5